Titre: Title:	Contrôleur dynamique pour l'accélération du logiciel GENCOL
Auteur: Author:	Hicham Lahlou
Date:	2002
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lahlou, H. (2002). Contrôleur dynamique pour l'accélération du logiciel GENCOL [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7008/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7008/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

CONTRÔLEUR DYNAMIQUE POUR L'ACCÉLÉRATION DU LOGICIEL GENCOL

HICHAM LAHLOU
DÉPARTEMENT DE MATHÉMATIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AVRIL 2002

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONTRÔLEUR DYNAMIQUE POUR
L'ACCÉLÉRATION DU LOGICIEL GENCOL

présenté par : LAHLOU Hicham

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. MOORE Marc, Ph.D., président

M. SOUMIS François, Ph.D., membre et directeur de recherche

M. TRÉPANIER Martin, Ph.D., membre

Remerciements

Je tiens à remercier mon directeur de recherche, François Soumis, qui m'a donné la chance de réaliser ce projet au sein du Groupe d'Études et de Recherche en Analyse des Décisions (GERAD). Il était toujours présent pour me donner les conseils et orientations dont j'avais besoin.

Je tiens à remercier également Daniel Villeneuve pour son soutien informatique sur le logiciel GENCOL. Son aide fût très appréciable.

Je tiens enfin à remercier Heykel Achour pour le soutien logistique qu'il m'a fourni tout au long de ma recherche.

Résumé

Le travail de ce mémoire a consisté à ajouter un contrôleur dynamique au logiciel d'optimisation GENCOL développé au GERAD.
GENCOL est un programme executable dont certains paramètres peuvent être modifiés afin de jouer sur la qualité de la solution et sur le temps de résolution.

Étant donné une limite finale connue sur le temps de résolution, le contrôleur fait varier les valeurs des paramètres internes de GENCOL pour l'atteindre.

Nous avons pu classer ces paramètres en trois catégories à savoir les paramètres du problème maître, du sous-problème et enfin les paramètres de branchement.

Nous avons ensuite étudié le comportement de la fonction objectif et du temps de résolution par rapport à une modification de ces paramètres en cours de résolution.

Nous avons développé une méthode d'estimation du temps de résolution d'un problème pendant son déroulement à partir d'une extrapolation de la décroissance du nombre de variables fractionnaires.

Les jeux de tests ont permis de montrer un bon comportement du contrôleur avec une fonction objectif à peine détériorée et un temps de résolution qui a été divisé par un facteur 20 dans certains cas. Avec un temps usager raisonnable, l'utilisation de quelques paramètres étudiés a permis de terminer les résolutions avec un écart inférieur à 10% entre le temps de calcul et le temps usager.

Une interface graphique a par ailleurs été développée afin de permettre à l'utilisateur de suivre le déroulement de la résolution.

Abstract

The work performed in this thesis consisted on adding a dynamical controller to the optimization software GENCOL developped in the GERAD.

GENCOL is a batch executable that contains some parameters that can be modified to tune the quality of the solution and the solution time.

Given a known final limit for the time to get a solution, the controller tunes some internal parameters of GENCOL to reach it.

We could classify these parameters in three groups which are the sub-problem parameters, the master problem parameters and the branch and bound parameters.

We then studied the behavior of the objective function and the solution time when modifying these parameters during the resolution.

The solution time of a problem is processed based on the decrease of the fractionnal variables.

The data tests confirmed a good behavior of the controller with a objective function close to the optimum and a solution time that was divided by a factor of 20 in some cases. With a reasonnable input time, the use of some studied parameters made the resolution end with a gap below 10% between the input time and the solution time.

A graphical interface was developped to allow the user to monitor the execution of the program GENCOL.

Table des matières

Liste des tableaux

Liste des figures

Introduction

Le plan d'opération des compagnies aériennes doit être souvent mis à jour du fait d'évènements imprévus appelés perturbations. Elles peuvent être liées aux mauvaises conditions météorologiques, aux problèmes mécaniques des avions provoquant des changements d'appareils planifiés aux vols, aux problèmes de contrôle aérien telle que la saturation des espaces aériens, à la circulation au sol (engorgement de l'aéroport en raison du trafic aérien anormalement élevé), et enfin, à l'absentéisme du personnel navigant.

La réactualisation des plans d'opération est une tâche difficile puisqu'il faut résoudre un problème mathématique complexe qui prend en compte de nombreuses variables et équations.

Il est inutile de préciser combien important cela peut être pour une compagnie aérienne d'être en mesure de résoudre cette réactualisation dans un court lapse de temps. En effet, chaque minute compte et peut être synonyme de perte d'argent pour la compagnie aérienne. Le logiciel d'optimisation utilisé est donc crucial pour réparer les perturbations dans des délais raisonnables.

De tous les logiciels d'optimisation présents sur le marché, peu d'entre eux sont en mesure de vous retourner un résultat dans un temps fixé par l'utilisateur. En effet, ces programmes prennent généralement en entrée un fichier décrivant le problème, le résolvent et retournent la solution optimale dans le temps nécessaire à la trouver. Cependant, le plupart des logiciels d'optimisation disposent de paramètres qui influent sur la qualité de la solution retournée et permettent aux usagers d'écourter quelque peu le temps de résolution mais ils n'ont pas moyen de connaître à l'avance le temps que cette résolution prendra.

Cette approche n'est pas forcément adaptée aux compagnies aériennes qui ont

besoin de réagir vite à des perturbations du plan d'opération et qui sont prêtes à faire quelques concessions sur l'optimalité de la solution pourvu que le temps de calcul soit bref.

Le sujet de cette maîtrise-recherche consiste à munir le logiciel d'optimisation GENCOL d'un contrôleur dynamique capable d'extrapoler le temps de résolution du problème et d'agir sur certains paramètres internes pour l'accélérer le cas échéant. Cette accélération a naturellement un coût à savoir une perte d'optimalité.

Dans le chapitre 1, nous allons définir plus en détails le problème qui nous est posé en énonçant les objectifs à rencontrer lors de la résolution du problème de la mise à jour des horaires d'équipage, problème sur lequel nous allons nous focaliser. Nous énoncerons par la suite les objectifs de recherche en définissant le sujet exact de ce mémoire.

Une revue de littérature sera faite dans le chapitre 2 dans laquelle nous décrirons plus en détail le logiciel d'optimisation GENCOL ainsi que la formulation mathématique des problèmes que celui-ci est destiné à résoudre. Nous mettrons l'emphase sur l'aspect mathématique du fonctionnement de ce logiciel en montrant que celui-ci résoud successivement un problème maître et un sous-problème. Nous préciserons ensuite que GENCOL est un logiciel modulaire avant de définir le rôle de chacun des modules qui le composent et leurs interactions. Nous définirons certains concepts importants que nous allons beaucoup utiliser pour accélérer la résolution d'un problème, comme la dominance puis nous définirons des termes mathématiques comme les étiquettes, le branchement...etc

Dans une deuxième partie, nous présenterons l'algorithme d'extrapolation de Levenberg-Marquardt que nous utiliserons pour extrapoler le temps de résolution du logiciel GENCOL.

Dans le chapitre 2, nous allons voir diverses façons d'accélérer la résolution d'un

problème. Nous classifierons ces façons en trois catégories selon qu'elles agissent sur le problème maître, le sous-problème ou le branchement. Par l'intermédiaire de nombreux jeux de tests, nous allons étudier le comportement de GENCOL à la modification de certains paramètres et tenter de poser les bases d'un contrôleur dynamique. Nous montrerons que l'optimalité est fortement détériorée par la modification de certains paramètres de GENCOL et chercherons les bonnes combinaisons de paramètres pour obtenir une accélération peu coûteuse en optimalité.

Dans le chapitre 3, nous définirons le fonctionnement d'un contrôleur dynamique capable d'accélérer la résolution de problèmes par le logiciel GENCOL. Nous baserons nos hypothèses sur les résultats obtenus au chapitre précédent puis établirons un système de priorités pour le contrôleur selon que le déroulement de la résolution est en accord avec le temps utilisateur. Nous montrerons quels paramètres il faudra modifier pour atteindre les objectifs de temps fixés.

Nous testerons le contrôleur dans le chapitre 4 sur des jeux de données provenant d'un problème typique d'horaires de chauffeurs d'autobus afin de s'assurer de la validité des théories énoncées dans les chapitres précédents.

Chapitre 1

Définition du problème

Tout en décrivant quelque peu les problèmes mathématiques auxquels sont confrontées les compagnies aériennes pour mettre à jour la planification horaire de ses appareils et équipages, ce chapitre définit le sujet de ce mémoire et met l'accent sur les diférentes étapes de recherche qui le constituent. Le problème d'horaire d' équipages dans le transport aérien est sujet à de nombreuses études [9] car il constitue un défi économique important pour les compagnies aériennes. En effet, on estime que les coûts d'équipage représentent une des parts les plus importantes des coûts d'opération directs et dépassent le milliard de dollars par an pour les grandes compagnies aériennes. La construction des horaires d'équipage est très complexe du fait de la taille des problèmes et de la complexité des conventions collectives et des lois gouvernementales qui régissent la profession.

Le problème d'horaire d'équipages peut se diviser en deux phases : la planification et la phase opérationnelle. La planification consiste à trouver d'abord un ensemble de rotations à partir d'un ensemble de segments de vols. Un segment de vol constitue un vol sans arrêt entre deux villes et une rotation représente une succession de segments de vols possiblement entrecoupée de repos nocturnes, commençant et se terminant à une base. Ensuite, il s'agit de construire des blocs mensuels personnalisés pour chaque membre d'équipage à partir des rotations obtenues précédemment.

Si la planification obtenue constitue généralement une bonne solution applicable, il n'en demeure pas moins qu'en pratique, les compagnies aériennes sont sujettes à des modifications de dernière minute qui les obligent à revoir cette planification. Ceci constitue la mise à jour des horaires d'équipage.

1.1 Le problème de mise à jour d'horaires d'équipages

Des perturbations surviennent fréquemment durant la phase opérationnelle. Ces perturbations peuvent être attribuées aux équipages (absence d'équipage, équipage en retard, etc.) ou au centre d'attribution des vols (révision d'horaire de vol, nouvelle route de vol, vol annulé, nouveaux vols, changement de type d'avion, etc.). Le bureau d'opération reçoit continuellement toutes les informations relatives aux changements dans les plans de vol et aux perturbations d'équipage.

Le problème de mise à jour des horaires d'équipage consiste à retourner une bonne solution au problème d'horaires d'équipages suite à une perturbation et ce dans les délais les plus brefs. Le problème d'horaires d'équipage consiste donc à modifier les blocs mensuels personnalisés de telle sorte que toutes les tâches (initiales et nouvelles) pour la période considérée soient couvertes à coût minimal avec les équipages disponibles, une tâche représentant le besoin d'un membre d'équipage pour un segment de vol. Ceci signifie qu'un segment de vol qui doit être couvert par n membres d'équipage définit n différentes tâches.

Les objectifs premiers lors de la modification des blocs mensuels personnalisés viennent comme suit :

- Couvrir toutes les tâches considérées dans le problème à coût minimum
- Demeurer aussi près que possible de la structure initiale des blocs mensuels

Les contraintes qui doivent être respectées au cours de la modification des blocs horaires personnalisés peuvent être divisées en trois groupes :

- Les contraintes relatives au problème de rotation : chaque nouvelle rotation doit être valide relativement à la convention collective et la règlementation gouvernementale.
- Les contraintes relatives à la construction des blocs mensuels personnalisés : un bloc mensuel personnalisé existant doit rester valide après le remplacement

d'une rotation par une autre.

- Les contraintes opérationnelles : ce sont les contraintes résultant du fait que certaines activités assignées à un membre d'équipage spécifique dans la phase de planification peuvent nécessiter le maintien de cette assignation. Parmi ces activités nous trouvons celles déjà débutées ou encore celles spécifiquement assignées à un membre d'équipage dans son bloc mensuel. Les contraintes opérationnelles peuvent également prendre en compte les contraintes relatives à l'utilisation des différents types de réserve.

La phase de planification pour le problème d'horaires d'équipage ne nécessitant qu'une résolution par mois, le temps de calcul n'est pas une forte contrainte pour les mathématiciens voués à la résolution du problème. Par contre lorsque des perturbations apparaissent dans la planification initiale, les opérateurs doivent prendre des décisions rapidement puisque ces perturbations se font peut-être déjà ressentir (absence non prévue d'un membre d'équipage, panne d'un appareil...). En conséquence, durant la phase opérationnelle, les opérateurs doivent réagir en temps réel et ne disposent pas toujours du temps nécessaire à l'élaboration d'une solution acceptable.

1.2 Objectif de recherche

Le problème de mise à jour d'horaires d'équipage dans sa phase opérationnelle représente des enjeux financiers importants pour les compagnies aériennes. En effet, les coûts de la moindre décision s'évaluent en milliers de dollars et il est donc important de ne pas commettre d'erreur sur les modifications à apporter à la planification.

Comme nous l'avons expliqué au paragraphe précédent, l'objectif est de couvrir toutes les tâches au moindre coût en s'éloignant le moins possible de la solution planifiée.

Le logiciel utilisé pour résoudre les problèmes énoncés est GENCOL (basé sur les principes mathématiques de génération de colonnes), développé au GERAD (Groupe

d'Études et de Recherche en Analyse des Décisions). Il permet de couvrir un ensemble de tâches avec des chemins satisfaisant des contraintes de ressources. Toutefois ce logiciel ne dispose pas de paramètre capable de limiter le temps de résolution d'un problème et le seul moyen d'accélérer quelque peu la résolution est d'utiliser un approximatif. Ceci consiste à limiter le domaine des solutions explorées. Cette manière de procéder permet effectivement de sauver quelques minutes de résolution mais d'une façon générale, ceci se fait au prix d'une forte perte en optimalité et il n'existe pas vraiment de plan de marche pour la résolution. Si l'on est chanceux, le résultat va être bon et beaucoup de temps de calcul aura été épargné mais cela peut se passer moins bien avec une mauvaise solution et un temps de résolution très grand.

L'objectif de cette maîtrise consiste à trouver une solution de qualité pour une résolution d'un problème d'horaires d'équipage en un temps prédéfini.

Pour arriver à cet objectif, une première étape va tout d'abord consister à se familiariser avec les différents paramètres de GENCOL qui permettent l'accélération de la résolution puis étudier l'effet de leur variation sur cette résolution. Ensuite, il va falloir trouver un moyen d'estimer l'avancement de la résolution lorsque GENCOL s'execute pour adapter la valeur des paramètres étudiés afin d'achever la résolution dans le temps prédéfini. Cette dernière étape va s'exécuter par l'intermédiaire d'un contrôleur dynamique qui va gérer en temps réel la valeur des paramètres en fonction de l'avancement de la résolution.

Chapitre 2

Littérature

Ce chapitre met l'emphase sur le logiciel GENCOL en décrivant dans un premier temps la formulation des problèmes mathématiques qu'il est susceptible de résoudre. Dans un second temps, nous détaillerons les fonctions des différents modules qui permettent la résolution de ces problèmes puis nous détaillerons l'aspect mathématique de l'algorithme d'extrapolation de Levenberg-Marquardt que nous utiliserons à la fin de ce mémoire pour évaluer le temps de résolution d'un problème.

2.1 GENCOL, un logiciel de génération de colonnes

Cette section décrit la formulation mathématique des problèmes résolus par GENCOL. Elle montre de quelle façon le logiciel procède pour atteindre une solution et décrit les différents modules qui le composent. Ce texte est essentiellement basé sur l'article de June Lavigne destiné à introduire le logiciel aux utilisateurs ([4]).

2.1.1 Aspect mathématique

GENCOL est destiné à résoudre des problèmes de couverture d'un ensemble de tâches modélisées par des chemins d'un graphe satisfaisants des contraintes de ressources.

La définition de ces problèmes est comme suit :

Considérons un ensemble de tâches qui sont associées à des noeuds et/ou des arcs sur un graphe. Ces tâches doivent être couvertes par des véhicules ou des équipages plus généralement appelés des commodités. Le problème considéré consiste à trouver le partitionnement à coût minimum de ces tâches ou chaque élément représente un chemin réalisable pour une commodité dans le graphe.

Soit K, l'ensemble des commodités, la commodité $k \in K$ se déplace le long d'un chemin dans le graphe $G^k = (V^k, A^k)$ où V^k et A^k sont respectivement l'ensemble des noeuds et des arcs. L'ensemble V^k est constitué de $N^k \cup \{o(k), d(k)\}$ où $o(k)$ et $d(k)$ sont les noeuds source et puits pour la commodité k. Le noeud source ne contient que des arcs sortants alors que le noeud puits ne contient que des arcs entrants. Pour chaque commodité $k \in K$, soit P^k l'ensemble des chemins réalisables dans G^k. Le coût du chemin $p \in P^k$ est noté c_p^k. Dans des applications simples, il correspond à la somme des coûts des arcs qui forment ce chemin. Toutefois, des versions adaptées de GENCOL sont capables de résoudre des applications dont la structure de coût est non-linéaire.

Soit W, l'ensemble des tâches qui doivent être couvertes. La variable θ_p^k représente le flot de la commodité $k \in K$ sur le chemin $p \in P^k$. Pour chaque chemin $p \in P^k, k \in K$, et chaque tâche $w \in W$, nous définirons un paramètre a_{wp}^k qui est égal à 1 si le chemin p contient la tache w et 0 sinon.

La formulation mathématique du problème de partitionnement est comme suit :

$$\min \sum_{k \in K} \sum_{p \in P^k} c_p^k \theta_p^k \tag{2.1}$$

sujet à

$$\sum_{k \in K} \sum_{p \in P^k} a_{wp}^k \theta_p^k = 1, \qquad \forall w \in W \tag{2.2}$$

$$\theta_p^k \geq 0, \qquad \forall p \in P^k, \forall k \in K \tag{2.3}$$

La fonction objectif (2.1) minimise le coût total de tous les chemins dans la solution. Les contraintes de partitionnement (2.2) indiquent que chaque tâche doit être incluse une seule fois dans un chemin. GENCOL est également capable de résoudre des problèmes où le membre de droite de cette équation est supérieur à 1. L'équation (2.3) impose que le flot sur chaque chemin soit non-négatif.

Contraintes d'intégrité :

Pour beaucoup de problèmes, il est nécessaire que les variables de chemin aient des valeurs entières. La contrainte (2.3) peut donc se réécrire :

$$\theta_p^k \in \{0,1\}, \forall p \in P^k, \qquad \forall k \in K \tag{2.4}$$

Contraintes globales :

Souvent des contraintes additionnelles qui lient plusieurs commodités sont nécessaires. Soit H l'ensemble de ces contraintes globales. Notons b_{hp}^k la contribution du chemin $p^k, k \in K$ à la contrainte globale $h \in H$ et b_h le membre de droite de cette contrainte. L'ensemble des contraintes globales est donné par l'équation suivante :

$$\sum_{k \in K} \sum_{p \in P^k} b_{hp}^k \theta_p^k \leq b_h, \qquad \forall h \in H \tag{2.5}$$

Agrégation de commodités :

D'habitude, chaque commodité représente une unité (par exemple, un véhicule ou un membre d'équipage). Toutefois, des commodités utilisant des graphes g^k identiques peuvent être groupées ensemble pour former des commodités agrégées. Les variables associées aux commodités identiques peuvent être confondues pour réduire le nombre de variables dans le problème et éviter des difficultés de symétrie. Si l'intégrité est requise sur les variables de chemin par le problème, l'équation (2.4) devient :

$$\theta_p^k entier, \qquad \forall p \in P^k, \forall k \in K \tag{2.6}$$

Sauf précisé, le reste de ce chapitre suppose le cas où les commodités ne sont pas aggrégées.

Contraintes de convexité :

Dans plusieurs applications, on a besoin de contraintes de convexité pour indiquer que chaque commodité doit être assignée à exactement un chemin. Soit $K^c \subseteq K$ l'ensemble des commodités sujettes à une telle contrainte. Les contraintes de convexité peuvent donc se réécrire :

$$\sum_{p \in P^k} \theta_p^k = 1, \qquad \forall k \in K^c \tag{2.7}$$

Une telle contrainte n'impose pas que la commodité correspondante k couvre des tâches puisqu'il est toujours possible de créer un chemin vide dans G^k, de $o(k)$ à $d(k)$ et qui représenterait la non utilisation de la commodité.

Variables supplémentaires :

Des variables additionnelles Y_s, $s \in S$ peuvent être nécessaires pour compléter la structure du problème . Soit a_{ws} et b_{hs} la contribution de la variable Y_s, $s \in S$, à la tâche $w \in W$ et à la contrainte $h \in H$, respectivement. Un coût c_s est associé à chaque variable y_s, $s \in S$. En considérant ces variables supplémentaires, la relaxation linéaire du problème s'écrit :

$$\min \sum_{k \in K} \sum_{p \in P^k} c_p^k \theta_p^k + \sum_{s \in S} c_s Y_s \tag{2.8}$$

sujet à

$$\sum_{k \in K} \sum_{p \in P^k} a_{wp}^k \theta_p^k + \sum_{s \in S} a_{ws} Y_s = 1, \qquad \forall w \in W \tag{2.9}$$

$$\sum_{k \in K} \sum_{p \in P^k} b_{hp}^k \theta_p^k + \sum_{s \in S} b_{hs} Y_s = b_h, \qquad \forall h \in H \tag{2.10}$$

$$\sum_{p \in P^k} \theta_p^k = 1, \qquad \forall k \in K^c \tag{2.11}$$

$$l_s \leq Y_s \leq u_s, \qquad \forall s \in S \tag{2.12}$$

$$\theta_p^k \geq 0, \qquad \forall p \in P^k, \forall k \in K \tag{2.13}$$

La nouvelle fonction objectif (2.8) inclut les coûts c_p^k associés aux chemins de G^k pour chaque commodité $k \in K$, ainsi que les coûts c_s associés aux variables supplémentaires. Les contraintes (2.12) limitent la valeur des variables supplémentaires. Dans certains cas, il peut être nécessaire que ces variables prennent des valeurs entières. La valeur des coefficients associés aux variables supplémentaires dépend de la définition de ces variables.

Contraintes locales :

Souvent, les chemins dans G^k, $k \in K$ le long desquels se déplacent les commodités sont sujets à certaines restrictions comme la capacité des véhicules, les intervalles de temps dans lesquels les tâches peuvent débuter, etc. Ces contraintes locales sont prises en compte durant la construction des chemins réalisables en utilisant des variables et des contraintes de ressources. Par exemple, dans un problème de tournée de véhicules où chaque tâche consiste à collecter les marchandises d'un client, la quantité de marchandise accumulée durant la tournée d'un véhicule est une ressource. La valeur de cette ressource est réactualisée après chaque visite d'un client. Des restrictions sur cette ressource doivent être imposées de façon à respecter la capacité du véhicule. Les consommations de ressources sont généralement assignées aux arcs d'un graphe alors que les restrictions sur les ressources sont assignées aux noeuds.

La mise à jour des horaires d'équipages :

Après avoir détaillé la formulation mathématique des problèmes que GENCOL est capable de résoudre, il est maintenant possible de faire un parallèle avec le problème de la mise à jour des horaires d'équipage ([5]). D'une manière générale, tous les problèmes d'horaires d'équipages consistent à trouver des horaires réalisables pour un ensemble de commodités qui couvrent un ensemble de tâches, ils s'énoncent comme suit :

Étant donnés un ensemble de commodités et un réseau associé à chaque com-

modité qui permet la génération de tous les horaires réalisables pour la commodité associée, trouver l'ensemble des horaires optimaux, un pour chaque commodité, de façon à ce que toutes les tâches soient couvertes un nombre de fois suffisant et que toutes les autres contraintes globales soient satisfaites.

Ces problèmes peuvent être modélisés grâce à une structure de réseau espace-temps où chaque noeud représente un lieu donné à un instant donné. Dans le cadre de la mise à jour des horaires, chaque commodité représente un membre d'équipage et les tâches représentent les fonctions de bord spécifiques sur tous les segments de vol. Chaque arc (i, j) représente une activité donnée qui commence au lieu et instant correspondant au noeud i et s'achevant au lieu et instant correspondant au noeud j. Dépendamment du problème, ces activités peuvent être des vols, des temps d'attente au sol, des nuits de repos, des journées de congé, etc. Un réseau est défini pour chaque commodité selon ses caractéristiques (localisation, disponibilités, qualifications, ...). Tous les horaires réalisables pour une commodité représentent les chemins du réseau associés à cette commodité. Toutes les contraintes restreignant le fait que les chemins soient réalisables sont prises en compte soit par la modélisation du réseau (temps de connexion minimum entre deux vols, nuits de repos consécutives, juxtaposition des activités...), soit lors de la construction d'un chemin à travers l'utilisation de variables de ressources qui permettent d'imposer des bornes inférieure et supérieure (nombre minimal de jours de repos pour une période donnée, nombres minimal et maximal d'heures de vol par période...). L'ensemble des commodités est aussi sujet à un ensemble de contraintes globales (contraintes sur les tâches couvertes, qualification minimale de l'équipage...).

2.1.2 Méthodologie de solution

La relaxation linéaire est résolue à l'aide d'une méthode de génération de colonnes. Cette manière de procéder résoud alternativement un problème maître restreint puis un ou plusieurs sous-problèmes. Ces sous-problèmes génèrent des che-

mins réalisables associés à des variables de chemin qui pourraient être ajoutées au problème maître restreint à l'itération suivante. Cette méthode de solution prévient l'énumération de toutes les variables de chemin possibles en ne considérant seulement qu'un sous-ensemble de ces variables et en générant d'autres lorsque nécessaire. Les variables de chemin et leurs coefficients correspondants sont également appelées colonnes dynamiques alors que les variables supplémentaires et leurs coefficients réciproques sont appelées colonnes statiques puisque ces colonnes sont toujours présentes dans le problème maître et ne peuvent être générées. Un nouveau problème de taille relativement petite, appelé problème maître restreint est associé à chaque sous-ensemble de variables. Le procédé de solution met en évidence un processus d'alternance entre le problème maître restreint et les sous-problèmes. À chaque itération le problème maître restreint est résolu pour un sous-ensemble de variables de chemin, les variables duales sont réactualisées et utilisées pour modifier le coût des arcs des sous-problèmes. Lorsqu'un chemin réalisable est trouvé par un sous-problème, une variable est associé à ce chemin dans le problème maître restreint et les coefficients de la colonne correspondante sont générés selon les contributions des noeuds et arcs trouvés le long de ce chemin. Le processus commence par résoudre le problème maître restreint composé de variables artificielles et de variables supplémentaires et s'achève lorsqu'aucun chemin de coût réduit négatif n'est généré par les sous-problèmes.

- **Problème maître restreint.** Le rôle du problème maître restreint consiste d'une part à trouver la solution optimale courante, c'est-à-dire en considérant les variables de chemin disponibles à cet instant et d'autre part à calculer la valeur des variables duales associées à cette solution. Ces variables duales seront utilisées pour calculer la fonction de coût qui sera elle-même utilisée lors de la résolution des sous-problèmes. Le problème maître restreint est un programme linéaire qui est résolu à l'aide d'un algorithme de simplexe primal ou dual dans GENCOL.
- **Sous-problèmes.** Le rôle des sous-problèmes consiste d'une part à vérifier si la solution courante est optimale pour le problème maître et d'autre part à générer des nouvelles variables de chemin avec des coûts réduits négatifs qui pourraient être ajoutées au problème maître restreint si la solution courante

n'est pas optimale. Un sous-problème différent est associé à chaque commodité $k \in K$. La condition d'optimalité est vérifiée en générant, pour chaque graphe $G^k, k \in K$, un chemin de coût réduit minimum. Si la valeur optimale de chaque sous-problème est non-négative alors la solution courante pour le problème maître restreint est optimale pour le problème maître. Dans le cas contraire, une ou plusieurs variables (associées à des chemins) obtenues sont ajoutées au problème maître restreint. Toutes les contraintes liées aux configurations de chemins réalisables (conservation de flot, limites sur les valeurs des ressources...) sont incluses dans chaque sous-problème. La fonction objectif d'un sous-problème est de telle sorte que sa valeur optimale correspond au coût réduit de la variable de chemin associée au chemin trouvé par le sous-problème. Chaque sous-problème est un problème de plus court chemin avec contrainte de ressource et est résolu à l'aide d'un algorithme de programmation dynamique ([2]). Cet algorithme permet de trouver non seulement le chemin de coût réduit minimum mais aussi d'autres chemins réalisables de coût réduit négatif qui pourraient être ajoutés au problème maître restreint.

- **Colonnes disjointes.** Les contraintes du programme résolu par GENCOL imposent que chaque tâche soit couverte exactement une fois dans la solution. On en déduit que deux chemins de la solution optimale ne peuvent pas contenir la même tâche. Cette simple observation a abouti à l'implémentation d'une stratégie de colonnes disjointes dans GENCOL. Cette stratégie a pour but de choisir parmi les variables de chemins réalisables générées par l'algorithme du sous-problème celles qui couvrent des tâches différentes et les ajouter dans le problème maître restreint à chaque itération. Pour chaque sous-problème résolu à une itération donnée, les colonnes générées sont groupées en blocs disjoints. Une tâche ne sera donc incluse qu'une fois dans un ensemble de chemins d'un bloc disjoint. Lorsque plusieurs sous-problèmes ont besoin d'être résolus, il est possible d'éliminer temporairement parmi les tâches des graphes qui ont été générées par les sous-problèmes précédents à l'itération courante, celles couvertes par des chemins associés à des colonnes présentes dans les premiers blocs disjoints. De cette façon, les sous-problèmes non encore résolus

ne peuvent pas générer des chemins couvrant ces tâches. Ces blocs disjoints peuvent également être utilisés pour établir une priorité entre les colonnes générées afin de réduire le nombre de variables de chemin ajoutées au problème maître restreint à chaque itération. Goffin et Vial ont démontré les fondements théoriques qui établissent le bien-fondé de cette méthode ([3]).

- **Modèles.** La recherche d'une borne inférieure requiert généralement un nombre important d'itérations impliquant chacune la résolution du problème maître restreint et de sous-problèmes tel qu'expliqué dans les paragraphes précédents. Plusieurs paramètres dans GENCOL contrôlent différents aspects de ce processus itératif. Un modèle se caractérise par un ensemble de paramètres qui influencent le déroulement de la résolution du processus itératif. Les modèles ont été créés afin de pouvoir faire évoluer dans le courant de la résolution la valeur de ces paramètres de façon statique et accélérer le processus de solution. Les modèles permettent de relaxer pour accélérer les itérations. On peut effectuer des relaxations au niveau du problème maître en agissant sur les critères d'arrêt ou encore sur les sous-problèmes en réduisant la taille du graphe ou en agissant sur les critères d'optimalité. Lors de cette résolution, un seul modèle est actif et les critères qui permettent de passer d'un modèle à l'autre sont nombreux et dépendent du choix de l'utilisateur. Celui-ci peut introduire autant de modèles qu'il le désire, si aucun modèle n'est précisé, GENCOL dispose d'un modèle optimal par défaut qui sera utilisé tout au long de la résolution et la valeur de tous les paramètres de modèle sera donc inchangée. Généralement une série de modèles peu contraignants et approximatifs sont résolus dans un premier temps pour approcher la solution optimale rapidement puis des modèles plus rigoureux et contraignants sont résolus pour compléter le processus de solution. Pratiquement, c'est l'expertise qui aide l'utilisateur dans ses choix de modèles. Certaines valeurs de paramètres sont particulièrement bien adaptées à un type de problème particulier et l'utilisateur pourra donc déterminer au préalable les modèles qui pourraient être efficaces pour le problème qu'il a à résoudre.

- **Solutions entières.** Si des contraintes d'intégralité sont imposées sur les variables de chemin ou les variables supplémentaires et que la solution optimale de la relaxation linéaire ne satisfait pas ces contraintes, le schéma de génération de colonnes doit être intégré dans un algorithme de branchement. Le processus de génération de colonnes est alors appliqué à chaque noeud de branchement pour déterminer la borne inférieure de ce noeud. Il existe plusieurs stratégies de branchement optimales et heuristiques disponibles dans GENCOL. Le processus de solution pour un problème donné peut utiliser une ou plusieurs de ces stratégies. À chaque noeud de branchement, les méthodes de branchement considérées sont mises en concurrence entre elles, on choisit le meilleur score pour déterminer la stratégie à utiliser. Parmi les méthodes de branchement les plus utilisées, on trouve la méthode *cfix* (diminutif de fixation de colonnes). Cette méthode consiste à fixer les valeurs d'un sous-ensemble de variables du problème maître. Lorsqu'une solution fractionnaire est identifiée (à la fin du processus de résolution associé à un noeud de branchement), toutes les variables du problème maître sont examinées de telle sorte qu'elles forment un ensemble de variables candidates avec des valeurs fractionnaires auxquelles on attribue un score entre 0.0 et 1.0 (de la moins bonne à la meilleure). Le score par défaut (il est possible de moduler cette fonction de score) correspond à la valeur décimale de la variable dans la solution courante. Un sous-ensemble de variable est alors formé en ne choisissant que les variables dont le score est supérieur ou égal à un seuil prédéterminé dans le respect des bornes inférieure et supérieure prédéfinies pour le cardinal de cet ensemble. La valeur des variables de ce sous-ensemble est alors fixée à la valeur du premier entier supérieur et les valeurs des colonnes correspondantes sont retirées des coefficients à droite de l'égalité du problème maître.

  D'autres méthodes de branchement sont également disponibles (itfix, bfix, sfix...) : elles consistent à fixer à 0 ou à 1 certaines variables représentant le flot sur les arcs du graphe. Toutes les méthodes de branchement peuvent être utilisées dans le même arbre d'énumération. L'exploration de cet arbre de recherche peut être effectuée selon trois politiques : profondeur d'abord,

meilleur d'abord ou une combinaison de ces deux politiques.

- **Variables de ressource.** Un ensemble de ressources R^k est associé à chaque commodité $k \in K$ de façon à modéliser les contraintes locales. L'état de ces ressources à chaque noeud $i \in V^k$ de chaque graphe $G^k, k \in K$, est représenté par le vecteur T_i^k des variables de ressource $T_i^{kr}, r \in R^k$. Une telle variable indique généralement la quantité de ressource $r \in R^k$ accumulée le long du chemin partiel p_i^k de la source $o(k)$ au noeud i si le noeud i est visité par la commodité k. $|R^k|$ intervalles de contraintes de la forme $[l_i^{kr}, u_i^{kr}]$ sont associés au noeud i et restreignent la valeur de T_i^{kr}. La consommation de la ressource r sur l'arc $(i,j) \in A^k$ est appelée t_{ij}^{kr}. Les variables de ressources sont nécessaires pour déterminer les chemins réalisables pour chaque graphe G^k.
- **Étiquettes.** Dans un algorithme de plus court chemin avec contraintes de ressource, une étiquette multi-dimensionnelle dénotée $E_i^k = (\overline{C}_i^k, T_i^k)$, est assignée à chaque chemin partiel p_i^k, où $\overline{C}_i^k$ représente le coût marginal de p_i^k. Cette étiquette est utilisée pour comparer p_i^k avec les autres chemins partiels s'achevant au noeud i. Par extension, une étiquette E_i^k associée à un chemin partiel réalisable est également dite réalisable et est de telle sorte que $T_i^{kr} \in [l_i^{kr}, u_i^{kr}], \forall r \in R^k$. Les étiquettes non réalisables sont évidemment retirées durant le processus de résolution. Puisque plusieurs chemins partiels réalisables peuvent s'achever à un même noeud, une liste d'étiquettes est associée à chaque noeud.
- **Dominance.** Comme mentionné précédemment, une liste d'étiquettes peut être associée à un noeud donné. Évidemment, tous les chemins partiels associés à ces étiquettes ne feront pas partie du plus court chemin. Dans un algorithme efficace de plus court chemin, une règle de dominance a été appliquée de façon à réduire la taille de la liste d'étiquettes à chaque noeud du graphe. La règle de dominance utilisée dans GENCOL est comme suit. Soient $E_{1i}^k = (\overline{C}_{1i}^k, T_{1i}^k)$ et $E_{2i}^k = (\overline{C}_{2i}^k, T_{2i}^k)$ deux étiquettes distinctes associées à des chemins s'achevant au noeud $i \in V^k$. La première étiquette domine la seconde, c'est-à-dire $E_{1i}^k \prec E_{2i}^k$, si et seulement si $\overline{C}_{1i}^k \leq \overline{C}_{2i}^k$ et $T_{1i}^k \leq T_{2i}^k, \forall r \in R^k$. Toutes les étiquettes dominées durant le processus de solution des sous-problèmes pour toute fonc-

tion d'extension non décroissante (cf. Desaulniers *et al.* (1998)) peuvent être éliminées.

Chaque ressource définie ajoute une dimension aux étiquettes utilisées dans l'algorithme de plus court chemin. Plus on met de ressources en dominance, plus la complexité de l'algorithme de plus court chemin croît, ainsi que le temps de résolution. D'une façon plus générale, plus on réduit le nombre de ressources, plus on réduit le nombre d'étiquettes conservées, plus on approxime mais il en résulte une accélération de la résolution. GENCOL permet à l'utilisateur d'utiliser des règles de dominance heuristiques qui permettent d'éliminer un grand nombre d'étiquettes. Ces règles comparent un sous-ensemble des composants des étiquettes plutôt que tous les composants. On peut donc utiliser la notion de modèle pour réduire le temps de résolution en employant dans un premier temps une série de modèles utilisant ces règles de dominance puis dans un second temps en appliquant un dernier modèle est qui comparerait tous les composants des étiquettes de façon à obtenir une bonne solution.

2.1.3 Implémentation modulaire de GENCOL

Puisque GENCOL est basé sur une technique de génération de colonnes intégrée dans un algorithme de branchement, le programme peut se décomposer en trois modules principaux. Le *MP* (pour Master Problem) destiné à résoudre la relaxation linéaire, le *CG* (pour Column Generation) destiné à résoudre les sous-problèmes et le *BB* (pour Branch and Bound) qui gère le processus de branchement. Cette séparation isole les différents algorithmes dans chacun des modules et permet aux programmeurs d'améliorer chaque module indépendamment. Comme la coordination entre le problème maître et les sous-problèmes requiert l'utilisation d'algorithmes complexes, un module additionnel appelé *SPP* (pour Shortest Path Problem) a été implémenté de façon à gérer efficacement la recherche d'une borne inférieure. La figure 2.1.3 illustre les relations entre chacun de ces modules du point de vue de la transmission de l'information. Ces modules reposent eux-mêmes sur des sous-

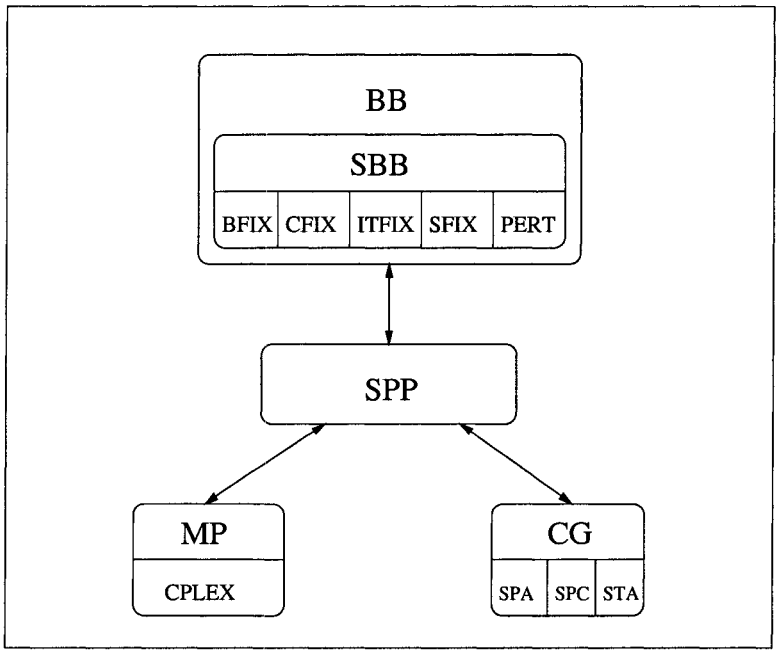

Figure 2.1 – Implémentation modulaire de GENCOL

modules qui contiennent différents algorithmes de solution. Le sous-module *CPLEX* consiste en une librairie de fonctions destinées à résoudre des programmes linéaires. *SPA,SPC,STA* sont différents algorithmes de plus court chemin, utilisés par *CG*. *SPA* est destiné à résoudre les graphes acycliques alors que *SPC* résoud les graphes cycliques. *STA* résoud plusieurs problèmes de plus court chemin en parallèle dans un graphe acyclique. Finalement, le module *BB* gère l'arbre de branchement alors que son sous-module *SBB* s'occupe des différentes stratégies de branchement.

- **Module SPP**

  Afin d'être complètement indépendant des algorithmes de branchement, ce module supporte tous les différents types de contraintes qui peuvent être imposés par les algorithmes de branchement. *SPP* inclut également un mécanisme capable de spécifier les contributions faites par les noeuds et arcs du réseau aux contraintes du problème maître. La réduction de la fenêtre de ressource, la simplification du réseau ainsi que les simplifications des contraintes du problème maître durant le processus de solution sont également traités par le *SPP*. Ce module supporte la notion de modèles utilisés pour gérer efficacement le processus itératif entre le problème maître et les sous-problèmes. *SPP* détermine

dans quel ordre et combien de sous-problèmes doivent être résolus à chaque itération. La coordination entre le problème maître et les sous-problèmes implique une supervision de la quantité d'information prise en charge par leur module correspondant de façon à contrôler la consommation de mémoire. Quand le procédé de solution du sous-problème l'autorise, *SPP* inclut des fonctions qui limitent le nombre d'étiquettes pour chaque noeud si nécessaire.

- **Module MP**

  Ce module constitue l'interface entre le solutionneur de programme linéaire (ici CLPEX) et les autres modules de GENCOL. Cette interface est nécessaire puisque CPLEX considère les coefficients associés aux variables du problème maître alors que les autres modules font référence aux chemins associés à ces variables.

- **Module CG**

  SPP gère la recherche d'une borne inférieure sans savoir quels algorithmes sont utilisés pour résoudre les problèmes de plus court chemin dérivant de ces sous-problèmes. Ces algorithmes constituent des sous-modules séparés. *CG* est une interface entre ces sous-modules et *SPP*. Cela permet aux utilisateurs d'implémenter des algorithmes spécialisés pour résoudre des sous-problèmes particuliers.

- **Module BB**

  Certaines règles pour gérer l'arbre de recherche sont présentes dans chaque algorithme de branchement. Un module générique *BB* qui gère toutes ces opérations a été implémenté. *BB* résoud un noeud de branchement, évalue la solution, évalue les différentes stratégies qui peuvent être utilisées pour ajouter des contraintes au problème, choisit le nouveau noeud de branchement et élimine les noeuds de branchement dominés. Ce module est également capable d'évaluer l'efficacité relative des différentes stratégies de branchement et de choisir la plus appropriée pour séparer le problème à un noeud de branchement donné. Toutes ces stratégies de branchement sont implémentées dans un sous-module *SBB*. Ce sous-module constitue la part non-générique d'un algorithme de branchement.

2.2 Algorithme de Levenberg-Marquardt d'extrapolation de fonctions non-linéaires

S'il existe de nombreuses méthodes pour effectuer une interpolation d'une fonction linéaire, nous disposons de peu d'outils pour résoudre le cas non linéaire. Marquardt a mis en avant une méthode élégante basée sur une ancienne suggestion de Levenberg. Basée sur les moindres carrés, cette routine fonctionne très bien en pratique et est devenue un standard des méthodes de moindres carrés non linéaires.

2.2.1 Les équations du problème

Considérons une interpolation dans le cas où nous disposons d'une fonction qui dépend de façon non linéaire de M paramètres inconnus $a_k, k = 1, 2, ..., M$. Il est possible de considérer une fonction mérite χ^2 et obtenir le meilleur jeu de paramètres en la minimisant. En mode non linéaire, cette minimisation doit être effectuée itérativement en utilisant une procédure qui s'arrêterait lorsque χ^2 cesserait de décroître. Proche du minimum, nous pouvons supposer que la fonction χ^2 peut bien s'approximer par une forme quadratique qui peut s'écrire :

$$\chi^2 \approx \gamma - d^t \cdot a + \frac{1}{2} a^t \cdot D \cdot a \tag{2.14}$$

où d est un vecteur de dimension M et D une matrice $M \times M$. Si l'approximation est bonne, nous savons comment à partir de la valeur courante des paramètres a_{cur} atteindre en un seul pas les valeurs recherchées a_{min} :

$$a_{min} = a_{cur} + D^{-1} \cdot [-\nabla \chi^2(a_{cur})] \tag{2.15}$$

Par contre, (2.14) peut être une approximation locale modeste à cause de la forme de la fonction que nous essayons de minimiser au point a_{cur}. Dans ce cas, nous devons faire un pas en utilisant la méthode de plus forte pente, en d'autres mots :

$$a_{next} = a_{cur} - k \times \nabla \chi^2(a_{cur}) \tag{2.16}$$

où k est une constante assez petite pour ne pas dépasser le point minimum dans la direction de descente. Pour utiliser (2.15) et (2.16), nous devons être capables de calculer le gradient de la fonction χ^2 pour tout jeu de paramètres a. De plus, pour utiliser (2.15), il faut connaître la matrice D qui est la matrice hessienne de la fonction mérite χ^2 pour tout a.

Dans le cas d'une interpolation non-linéaire, il est possible de conaître ces valeurs dans la mesure où nous connaissons la forme exacte de χ^2 puisqu'elle est basée sur le modèle de la fonction que nous avons nous même spécifiée. Nous pouvons donc utiliser à notre guise (2.15) et la seule raison d'utiliser (2.16) serait un échec de (2.15) qui signifierait une mauvaise approximation de locale de (2.14).

Le modèle à interpoler est de la forme :

$$y = y(x; a) \tag{2.17}$$

La fonction mérite s'écrit alors :

$$\chi^2(a) = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i; a)}{\sigma_i} \right)^2 \tag{2.18}$$

Dans l'équation précédente, σ_i représente la déviation standard pour la valeur y_i correspondante. Le gradient de χ^2 par rapport aux paramètres a, qui s'annule au minimum de χ^2, s'écrit :

$$\frac{\partial \chi^2}{\partial a_k} = -2 \sum_{i=1}^{N} \frac{[y_i - y(x_i; a)]}{\sigma_i^2} \frac{\partial y(x_i; a)}{\partial a_k}, \qquad k = 1, 2, ..., M \tag{2.19}$$

En ajoutant une dérivée partielle additionnelle, cela donne :

$$\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i; a)}{\partial a_k} \frac{\partial y(x_i; a)}{\partial a_l} - [y_i - y(x_i; a)] \frac{\partial^2 y(x_i; a)}{\partial a_k \partial a_l} \right] \tag{2.20}$$

Par convention nous enlevons les facteurs 2 en définissant :

$$\beta_k \equiv -\frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_k} \qquad \alpha_{kl} \equiv \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_k \partial a_l} \tag{2.21}$$

obtenant ainsi $[\alpha] = \frac{1}{2} D$ dans l'équation (2.15), ce qui fait que cette équation peut être réécrite comme un ensemble d'équations linéaires :

$$\sum_{l=1}^{M} \alpha_{kl} \delta_{a_l} = \beta_k \tag{2.22}$$

Ces équations sont résolues pour des incréments de δa_l qui, ajoutés à l'approximation courante nous donne l'approximation suivante.
L'équation (2.16) de plus forte descente se traduit :

$$\delta_{a_l} = constant \times \beta_l \tag{2.23}$$

Notons que les composantes de la matrice hessienne α_{kl} (2.20) dépendent non seulement de la dérivée première mais aussi de la dérivée seconde des fonctions de base par rapport à leurs paramètres. Des simplifications mathématiques nous permettent de ne considérer que les dérivées premières ([6]), ce qui nous donne :

$$\alpha_{kl} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i; a)}{\partial a_k} \frac{\partial y(x_i; a)}{\partial a_l} \right] \tag{2.24}$$

Il faut bien comprendre ici que la condition lorsque χ^2 est minimum, $\beta_k = 0$ pour tout k est indépendante de la façon avec laquelle $[\alpha]$ est définie.

2.2.2 La méthode de Levenberg-Marquardt

Marquardt a proposé une méthode intéressante basée sur des reflexions de Levenberg ([7]) qui propose des alternances entre la méthode des extrêmes de l'inverse du Hessien (2.22) et la méthode de plus forte descente (2.23). Cette dernière méthode est utilisée loin du minimum alternant de façon continue avec la première méthode lorsque le minimum est proche. Cette méthode marche très bien en pratique et est devenue un standard des méthodes de moindres carrés.

Cette méthode est basée sur deux remarques élémentaires qui ont leur importance. Considérons la constante de l'équation (2.23). Il n'y a aucune information sur sa valeur ou au moins son ordre de grandeur dans le gradient. Celui-ci nous renseigne seulement sur la pente et non sur le pas. La première remarque de Marquardt réside dans le fait que les composantes de la matrice hessienne, qui ne sont pas vraiment utilisables, peuvent cependant renseigner sur l'ordre de grandeur du problème.

La quantité χ^2 est sans dimension alors que β_k a la dimension de $\frac{1}{a_k}$. La constante de proportionnalité entre β_k et δ_{a_k} doit donc avoir la dimension de a_k^2. En parcourant les composants de $[\alpha]$, on se rend compte qu'il n'y a qu'une quantité avec ces dimensions, il s'agit de $1/\alpha_{kk}$, l'inverse des éléments diagonaux de la matrice, ce qui fixe l'échelle de la constante. Mais il se peut que cette échelle soit trop grande , il faut alors diviser la constante par un facteur λ avec la possibilité de fixer $\lambda >> 1$ pour faire un pas efficace. En d'autres mots, remplaçons l'équation (2.23) par :

$$\begin{aligned} \delta_{a_l} &= \frac{1}{\lambda\alpha_{ll}}\beta_l \\ \lambda\alpha_{ll}\delta_{a_l} &= \beta_l \end{aligned} \tag{2.25}$$

Les a_{ll} sont tous positifs du fait de l'équation (2.24).
La seconde remarque de Marquardt est que les équations (2.22) et (2.25) peuvent être combinées si nous définissons une nouvelle matrice $[\alpha']$ de la façon suivante :

$$\alpha'_{jj} \equiv \alpha_{jj}(1+\lambda) \tag{2.26}$$

$$\alpha'_{jk} \equiv \alpha_{jk} \tag{2.27}$$

Ces équations remplacent les équations (2.22) et (2.25) par :

$$\sum_{l=1}^{M} \alpha'_{kl}\delta_{a_l} = \beta_k \tag{2.28}$$

Lorsque λ est très grand, la matrice $[\alpha']$ est forcée à être diagonalement dominante, l'équation (2.28) va devenir identique à l'une des équations (2.25). Par contre lorsque λ s'approche de 0, (2.28) tend à se rapprocher de (2.22).
Avec des valeurs initiales quelconques pour les paramètres a, l'algorithme de Marquardt s'exprime comme suit :

- Calculer $\chi^2(a)$

- Prendre une faible valeur pour λ, par exemple $\lambda = 0.001$

- Étape 1 : Résoudre les équations linéaires (2.28) pour δa et évaluer $\chi^2(a+\delta a)$

- Si $\chi^2(a+\delta a) \geq \chi^2(a)$, augmenter λ d'un facteur 10 et retourner à l'étape 1.

- Si $\chi^2(a + \delta a) < \chi^2(a)$ diminuer λ d'un facteur 10, réactualiser $a = a + \delta a$ et retourner à l'étape 1

La condition d'arrêt de ces itérations se caractérise par la faible décroissance de χ^2 d'une itération à l'autre.

Chapitre 3

Accélération de la résolution

Comme nous l'avons précisé dans le chapitre 2, la première étape de la reflexion aboutissant à l'écriture d'un contrôleur pour GENCOL nécessite la parfaite connaissance de l'ensemble des paramètres du logiciel susceptibles d'être modifiés afin d'accélérer une résolution. Nous allons maintenant présenter les trois différentes champs de recherche dont nous disposons pour tenter d'accélérer le problème.

3.1 Les paramètres du sous-problème

Le principal champ de recherche que nous allons évoquer pour faire évoluer le temps de résolution en utilisant les paramètres du sous-problème est lié à la dominance sur les ressources.

3.1.1 La dominance sur les ressources

Explication à travers un exemple

La dominance est une méthode d'élimination d'étiquettes qui, à chaque noeud du graphe, va garder les colonnes (ou chemins) susceptibles d'être optimales et ce par comparaison des valeurs des coûts et des ressources des différentes étiquettes présentes à ce noeud. Si on effectue une élimination basée sur toutes les ressources, on garde nécessairement la solution optimale ([8]). Sur la figure (3.1), on peut voir la valeur des étiquettes présentes aux noeuds N1 et N2 qui sont transférées au noeud N3

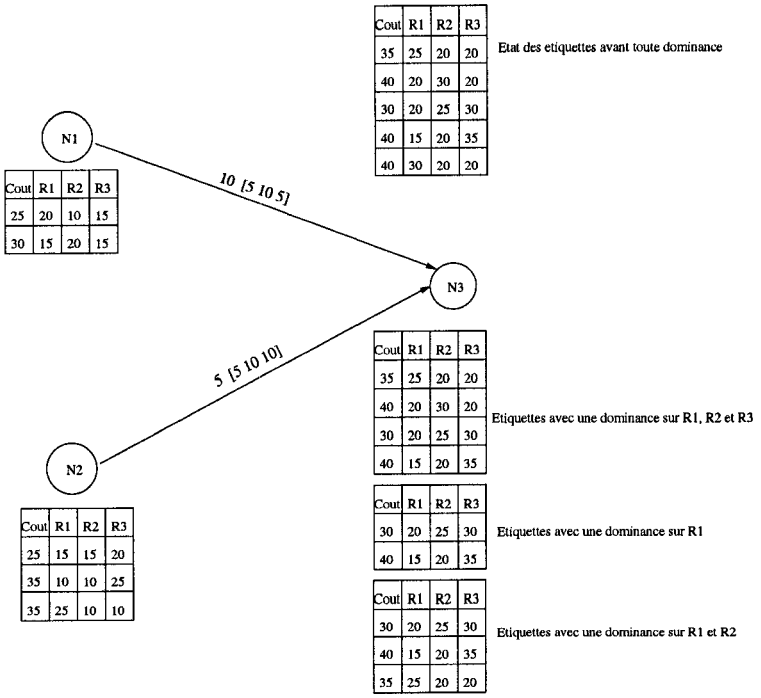

Figure 3.1 – La dominance par l'exemple.

avec ajustement des valeurs des coûts et ressources en fonction de l'arc emprunté. À titre d'exemple, nous avons montré les différents résultats en fonction des ressources mises en dominance au noeud N3. Si seule la première ressource R1 est mise en dominance, seulement deux chemins seront conservés car ils dominent par rapport à la première ressource (et implicitement au coût) et deux chemins seront éliminés. Il est évident que dominer sur une ressource va éliminer beaucoup de colonnes et ne garantit pas l'optimalité de la solution car certaines colonnes éliminées peuvent tout de même être intéressantes par rapport aux autres ressources non dominées. Toutefois, dominer sur une ressource aboutit à une solution dans des temps bien moindres qu'avec une dominance sur l'ensemble des ressources mais ceci au prix d'une dégradation de la solution de la fonction objectif. Le choix de la ressource à mettre en dominance a une influence très nette sur la valeur de la fonction objectif, il est plus intéressant pour ne pas trop la dégrader de mettre en dominance une ressource contraignante pour le problème.

Lorsque deux ressources sont mises en dominance, on constate (toujours sur

notre exemple) qu'un chemin n'est plus dominé, on a donc élargi le champ des colonnes renvoyées au problème maître. Lorsque l'on domine sur les trois ressources du problème, une colonne sur toutes celles possibles est éliminée. Puisque la dominance s'effectue sur toutes les ressources du problème, cette colonne ne peut faire partie de la solution.

Pour récapituler, moins il y a de ressources en dominance, plus la solution du problème sera dégradée mais elle sera rendue dans des temps plus brefs. En effet, enlever une ressource revient à contraindre le problème. Mettre en dominance toutes les ressources assure un maximum d'optimalité de la solution car seules les colonnes dominées sur leur coût et l'ensemble de leurs ressources seront éliminées.
Pour être tout à fait rigoureux, nous pouvons préciser que la dominance n'est pas effectuée sur le coût comme le préconise la figure (3.1) mais sur le coût réduit. En effet, c'est plus le caractère attractif d'une colonne qui nous intéresse plutôt que son coût en tant que tel.

3.1.2 Dominance sur le noeud puits

Dans GENCOL, il existe un paramètre de modèle nommé *SpaModResDomFlow* qui prend en arguments les noms des ressources à mettre en dominance pour l'ensemble des noeuds du graphe exception faite du puits. En effet, Gencol permet de faire la distinction entre le puits et tous les autres noeuds du graphe à travers la valeur du paramètre de modèle *SpaModResDomSinkUse*. Si sa valeur est fixée à 1, alors les paramètres de dominance du noeud puits seront lus dans SpaModResDomSink, si elle est fixée à 0, il n'y a pas de dominance effectuée sur le puits.

Ne pas effectuer de dominance sur le noeud puits nous permet de sauver le temps de calcul de cette dominance mais en contrepartie, on accroît le nombre d'étiquettes conservées au noeud puits et donc le nombre de colonnes renvoyées à chaque itération au problème maître.

Selon le type du problème, il peut être intéressant de ne pas effectuer cette dominance sur le noeud puits. En effet les sous-problèmes retournent au problème maître certaines colonnes qui sont certes dominées mais qui peuvent être profitables dans la mesure où elles couvrent possiblement d'autres tâches que les colonnes qui les dominent. De ce fait, elles enrichissent l'ensemble des colonnes du problème maître et écourtent le nombre d'itérations, économisant ainsi du temps.

Théoriquement, si toutes les ressources sont mises en dominance, que l'on domine ou non sur le noeud puits n'influence pas la solution finale puisque toutes les colonnes qui feront partie de la solution se retrouveront identiquement dans la base du problème maître. Par contre, si toutes les ressources ne sont pas mises en dominance, il se peut que le résultat diffère, puisqu'il y a de fortes chances que la valeur finale de la fonction objectif ne soit pas celle de l'optimum et que les deux résolutions empruntent des chemins différents pour explorer l'arbre de branchement.

Nous pouvons donc fortement accélérer la résolution d'un problème sous GENCOL et ce en augmentant le nombre de colonnes à coût réduit négatif qui sont envoyées dans le problème maître.

Si le fait de ne pas dominer sur le noeud puits et d'envoyer plus de colonnes au problème maître accélère le temps de résolution, nous avons pensé désactiver cette dominance sur l'ensemble des noeuds qui ont un arc vers un noeud puits. De cette façon, nous allons non seulement envoyer au problème maître l'ensemble des colonnes qui sont dominées au noeud puits mais aussi celles qui sont dominées à un des noeuds qui précèdent le puits.

Pour ce faire, il suffit de changer dans *spa.c* du répertoire *spp* de GENCOL, la ligne qui stipule que la dominance ne doit pas être efectée sur le noeud puits par une ligne qui stipule que la dominance ne doit pas être effectuée sur le noeud puits et ses prédecesseurs.
Après de nombreux jeux de tests, nous avons observé que dans à peu près la moitié

des cas, le problème s'en trouvait accéléré mais que dans le reste des cas, le temps de résolution s'allongeait. Nous avons préféré abandonner l'idée.

Dans le paragraphe suivant, nous allons présenter des résultats numériques obtenus à paertir de jeux de tests.

3.1.3 Résultats numériques

Le tableau (3.1) résume les résultats obtenus lors de la résolution d'un problème comprenant 4 ressources et 349 tâches à couvrir. La légende de ce tableau vient

Tableau 3.1 – Résultats numériques sur un problème à 4 ressources

		Relaxation linéaire (Noeud 0)				**Problème complet**			
R	Dom. puits	**FO**	SP	PM	SP+PM	**FO**	SP	PM	SP+PM
4	Oui	42850	847.4	192.4	1039.8	43300	2651.0	288.8	2939.8
4	Non	42850	407.9	79.5	487.4	43300	1977.2	147.1	2124.3
3	Oui	42850	326.2	126.5	452.7	43300	1386.0	234.2	1620.2
3	Non	42850	247.5	95.7	343.2	43300	1464.0	212.9	1676.9
2	Oui	42850	579.1	257.0	836.1	43800	1630.9	351.3	1982.2
2	Non	42850	229.1	88.4	317.5	43600	1097.6	186.0	1283.6
1	Oui	42850	478.7	250.9	729.6	43300	1313.0	363.3	1676.3
1	Non	42850	203.3	90.4	293.7	43400	973.1	201.4	1174.5

comme suit :

R : Nombre de ressources utilisées.

Dom. puits : La domination sur le puits est-elle activée ?

FO : Valeur de la fonction objectif finale.

SP : Temps en secondes pour résoudre les itérations du sous problème .

PM : Temps en secondes pour résoudre les itérations du problème maître.

SP+PM : Somme de SP et PM.

Il est intéressant de comparer sur les différents tests effectués le nombre d'itérations à divers moments du problème ainsi que le temps moyen par itération. Ces données sont regroupées dans le tableau (3.2) :

Tableau 3.2 – Résultats numériques par itération sur un problème à 4 ressources

		Relaxation linéaire					**Problème complet**					
R	Dom	**FO**	Iter	SP/i	PM/i	Tps/i	**FO**	Iter	NB	SP/i	PM/i	Tps/i
4	Oui	42850	164	5.17	1.17	6.34	43300	648	48	4.09	0.45	4.54
4	Non	42850	80	5.1	0.99	6.09	43300	549	53	3.6	0.27	3.87
3	Oui	42850	110	2.97	1.15	4.12	43300	629	52	2.2	0.37	2.57
3	Non	42850	85	2.91	1.13	4.04	43300	675	51	2.17	0.32	2.49
2	Oui	42850	205	2.82	1.25	4.07	43800	729	51	2.24	0.48	2.72
2	Non	42850	85	2.7	1.04	3.74	43600	533	47	2.06	0.35	2.41
1	Oui	42850	207	2.31	1.21	3.52	43300	671	47	1.96	0.54	2.5
1	Non	42850	85	2.34	1.04	3.38	43400	523	49	1.86	0.39	2.25

La légende de ce tableau vient comme suit :

R : Nombre de ressources utilisées.

Dom. : La domination sur le puits est-elle activée ?

FO : valeur de la fonction objectif.

Iter : Nombre d'itérations nécessaires à la résolution de la relaxation linéaire.

SP/i. : Temps moyen en seconde nécessaire à la résolution du sous-problème pour une itération.

PM/i. : Temps moyen en seconde nécessaire à la résolution du problème maître pour une itération.

Tps/i. : Temps moyen en secondes pour résoudre une itération.

NB : Nombre de noeuds de branchement.

Commentaires sur la dominance pour un sous-ensemble de ressources

La première remarque importante sur les résultats numériques obtenus réside dans le gain de temps non négligeable entre les problèmes au cours desquels est effectuée une dominance sur le noeud puits et leur équivalent où cette dominance

n'est pas appliquée. La principale raison de ce gain de temps réside dans la forte diminution du nombre d'itérations. Pour la relaxation linéaire, les résultats sont particulièrement bons puisque l'on parvient à diviser le temps de résolution par deux. Pour le problème complet, généralement les temps sont réduits mais de façon moins significative et on constate beaucoup de variance pour le nombre de noeuds de branchement. Pour le cas à trois ressources, le nombre d'itérations est même augmenté en n'appliquant pas de dominance sur le noeud puits et il en résulte un accroissement du temps pour résoudre le problème. De plus retirer des ressources en dominance accélère fortement la résolution du problème comme nous l'avons expliqué précédemment mais ceci au prix d'une perte dans la valeur finale de la fonction objectif, bien qu'ici cette perte est de l'ordre du pourcent.

La diminution du nombre total d'itérations est une conséquence du fait qu'à chaque itération, plus de colonnes sont envoyées au problème maître dans le cas où on ne domine pas sur le noeud puits. Parmi ces colonnes, certaines sont dominées et auraient dû être éliminées. Toutefois, ces colonnes sont réalisables puisqu'elles se rendent jusqu'au noeud puits et bien qu'elles sont dominées, il se peut qu'elles le soient par rapport à des colonnes qui ne couvrent pas les mêmes tâches. Ces colonnes sont donc potentiellement intéressantes et n'auront pas à être générées lors d'itérations futures. Il en résulte une diminution du nombre total d'itérations.

Il faut bien comprendre que les hypothèses qui sont émises jusque là sont liées à un déroulement spécifique et intrinsèque de résolution et il se peut que dépendamment du problème, ne pas dominer sur le noeud puits entrave plus les choses que cela ne les améliore comme le montre le cas à trois ressources. En fait, on n'économise pas beaucoup de temps dans l'exploration de l'arbre de branchement et la variance qui règne dans le nombre de noeuds de branchement peut faire augmenter une durée de résolution.

Si la valeur moyenne du temps par itération pour les sous-problèmes n'est guère modifiée lorsqu'à ressource constante, on domine ou non sur le noeud puits, on peut

se demander pourquoi, lorsqu'on ne domine pas, le temps moyen par itération du problème maître se trouve accéléré alors que l'on est supposé renvoyer plus de colonnes au problème maître. La figure (3.2) nous montre en pointillés le nombre de

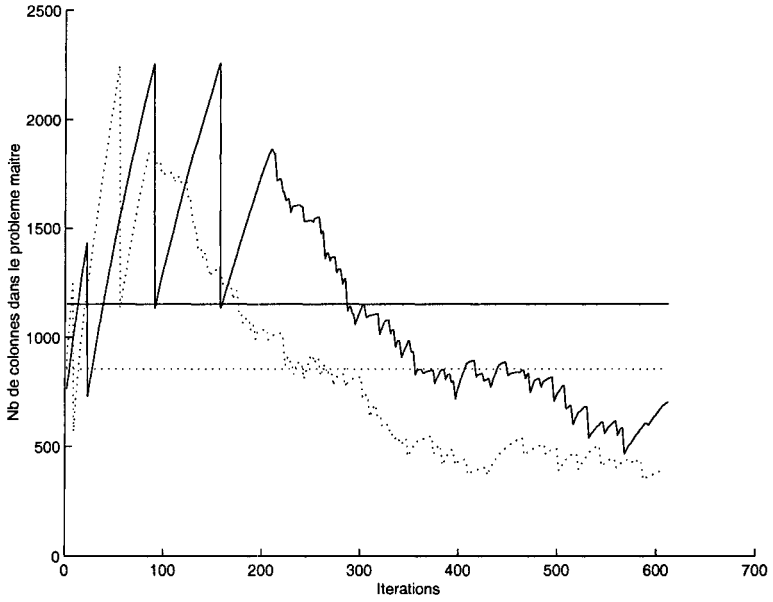

Figure 3.2 – Évolution du nombre de colonnes dans le problème maître en fonction du nombre d'itérations.

colonnes dans le problème maître au cours de la résolution lorsque l'on ne domine pas sur le noeud puits alors que le trait plein représente le cas où l'on domine sur le noeud puits. Les droites horizontales représentent le nombre moyen de colonnes dans le problème maître pendant le déroulement de la résolution. On voit qu'en moyenne, il y a moins de colonnes dans le problème maître lorsque l'on ne domine pas sur le noeud puits. Ceci résulte du fait que la relaxation linéaire est résolue plus rapidement dans le cas où l'on ne domine pas sur le noeud puits et le branchement est donc entrepris plus rapidement éliminant ainsi de nombreuses colonnes du problème maître. Effectivement, d'un noeud de branchement à l'autre, une colonne est fixée (pour la méthode de branchement Cfix utilisée pour ces jeux de tests) et de nombreuses colonnes présentes dans le problème maître deviennent soit inutiles, soit irréalisables et ce soit parce qu'elles couvrent une des tâches déjà couvertes par la colonne fixée, soit parce qu'elles ne satisfont plus les nouvelles contraintes revues et corrigées du fait de la fixation de la colonne. Pour le cas où on ne domine pas sur

le noeud puits, plus de colonnes sont retirées d'un noeud de branchement à l'autre, et en moyenne il en reste moins dans le problème maître comparé au cas où on l'on domine.

3.2 Une accélération basée sur un critère d'arrêt

Dans GENCOL, l'exécution d'un modèle prend fin lorsqu'aucune colonne à coût réduit négatif n'est générée. Toutefois, il existe plusieurs paramètres qui permettent de définir un critère d'arrêt pour stopper prématurément un modèle. Si plusieurs modèles sont définis par l'usager, GENCOL continuera la résolution du problème avec le modèle suivant. Si seul un modèle est défini, GENCOL cesse la résolution du noeud de branchement courant et passe au noeud suivant. On estime que dans certains cas, stopper prématurément la résolution d'un noeud de branchement n'engendre pas des pertes significatives pour la solution du problème et permet d'économiser de précieux instants.

3.2.1 La décroissance de la fonction objectif comme critère

Parmi ces critères d'arrêt, certains fonctionnent sur la décroissance de la fonction objectif sur un tronçon d'itérations. Le paramètre de modèle *SppModLogItr* définit la longueur du tronçon. Si nous notons c_d la valeur de la fonction objectif à la dernière itération, c_p la valeur de la fonction objectif à la première itération du tronçon, le modèle prend fin si :
$SppModLogItr \geq 2$ et $c_p - c_d \leq SppModLogCostDecMin$
où *SppModLogCostDecMin* est le paramètre de modèle défini par l'usager qui représente la décroissance minimale que doit avoir la fonction objectif sur un tronçon de *SppModLogItr* itérations pour poursuivre le déroulement de la résolution du modèle. On peut signaler ici qu'il existe également un paramètre semblable à *SppModLogCost-*

DecMin à la différence près qu'il est relatif à la valeur de la fonction objectif (l'usager impose un pourcentage de décroissance minimal) mais il ne sera pas étudié dans ce rapport.

Avantages et inconvénients

Le seul critère d'arrêt de modèle est par défaut le fait qu'aucune colonne de coût réduit négatif ne soit générée par les sous-problèmes. Pour de nombreux problèmes, au fur et à mesure du déroulement de la génération de colonnes, de moins en moins de colonnes sont introduites dans le problème maître et cela a souvent pour conséquence un ralentissement de la décroissance de la fonction objectif. Il n'est pas rare qu'avant de ne plus générer de colonnes à coût réduit négatif, la fonction objectif stagne à une même valeur ou décroît très lentement pendant de nombreuses itérations coûteuses en temps, c'est ce que nous nommerons dans ce mémoire les queues de résolution des noeuds de branchement. L'objectif de ce paragraphe est de montrer comment utiliser le paramètre *SppModLogCostDecMin* afin d'éviter que les itérations de génération de colonnes ne s'éternisent avec peu de modification de la valeur de la fonction objectif. Bien entendu, il va en résulter un gain non négligeable en temps de résolution, par contre il se peut que l'absence des colonnes qui auraient dû être générées dans des itérations futures (et qui ne seront donc pas dans le problème maître) porte atteinte à la valeur de la fonction objectif finale. En effet, cela peut avoir entre autre une influence sur la ou les colonnes fixées par la méthode de branchement *Cfix* ou tout simplement la valeur finale pour le modèle en cours n'était pas encore atteinte alors que la décision d'interrompre le modèle a été prise.

Variation de la longueur du tronçon étudié avec une valeur de décroissance minimale de la fonction objectif positif

Dans les exemples numériques qui vont suivre, nous allons voir qu'il existe deux façons différentes d'utiliser le paramètre de modèle *SppModLogCostDecMin*. Nous pouvons soit modifier la longueur du tronçon sur lequel nous étudions la décroissance de la fonction objectif pour un paramètre *SppModLogCostDecMin* fixé, soit pour une longueur *SppModLogItr* de tronçon fixée faire varier la valeur de *SppModLogCostDecMin*. Ces deux méthodes nous permettent d'envisager l'utilisation de ces paramètres de deux approches différentes. Dans les tests numériques, nous avons intégré la notion de dominance sur les ressources comme expliqué au paragraphe précédent sur le même jeu de test qui comprend quatre ressources. Ce problème est lancé avec une banque d'un seul modèle, ce qui signifie que l'interruption du modèle est synonyme d'interruption de la résolution du noeud de branchement courant.

Tableau 3.3 – Effet du paramètre *SppModLogItr* pour une valeur de *SppModLogCostDecMin* nulle sur la résolution d'un problème par GENCOL

R	Dom.	*SppModLogItr*	NB	Nds coupés	Nb Itr	SP	PM	SP+PM	FO
4	non	5	55	29	515	1911.8	237.9	2149.7	44800
4	non	4	60	26	549	1978.7	227.0	2205.7	48900
4	non	3	57	38	432	1564.9	220.8	1785.7	51400
4	non	2	57	56	380	1395.4	217.9	1613.3	45200
3	non	5	55	18	470	1001.1	229.0	1230.1	43900
3	non	4	54	28	464	993.4	222.8	1216.2	44700
3	non	3	50	41	385	821.3	223.2	1044.5	43400
3	non	2	53	48	374	804.7	253.3	1058.0	49100
2	non	5	48	23	469	978.3	241.6	1219.9	43300
2	non	4	51	38	491	1051.7	264.9	1316.6	45100
2	non	3	53	36	443	897.5	232.4	1129.9	44700
2	non	2	56	46	390	778	256.6	1034.6	47800
1	non	5	49	25	464	868.3	229.2	1097.5	43300
1	non	4	52	33	479	884.6	227.2	1112.3	43400
1	non	3	51	35	416	768.3	223.8	992.1	43500
1	non	2	56	55	374	690.4	228.3	918.7	44600

La légende de ce tableau se lit comme suit :

R : Nombre de ressource mises en dominance.

Dom : La dominance sur le noeud puits est-elle effectée ?

SppModLogItr : Valeur de *SppModLogItr*.

NB : Nombre de noeuds de branchement nécessaires à la résolution du problème.

Nds coupés : Nombre de noeuds de branchement où la résolution s'est vue interrompue du fait d'une décroissance nulle de la fonction objectif pour *SppModLogItr* itérations.

Nb Itr : Nombre d'itérations nécessaires à la résolution du problème.

SP : Temps pour résoudre les sous-problèmes.

PM : Temps pour résoudre le problème maître.

SP + PM : Temps nécessaire pour résoudre le problème.

FO : Valeur de la fonction objectif finale.

Des chiffres du tableau (3.3), on peut faire plusieurs remarques intéressantes. Tout d'abord, nous constatons que la valeur finale de la fonction objectif rendue par GENCOL se détériore avec la diminution de la valeur du paramètre *SppModLogItr*. Ceci est logique puisqu'en n'autorisant que très peu d'itérations pour que la fonction objectif décroisse, on contraint fortement le problème. Dans le cas où *SppModLogItr* vaut deux, cela signifie que si les colonnes de coût réduit négatif introduites dans le problème maître à l'itération courante ne parviennent pas à faire diminuer la valeur de la fonction objectif, le modèle est stoppé et on explore un autre noeud de branchement. Toutefois, rien ne prouve que la valeur finale du noeud de branchement a été atteinte puisqu'il reste encore probablement d'autres colonnes à coût réduit négatif qui n'ont pas été introduites dans le problème maître. Un autre phénomène intéressant à constater ici est la relation entre le nombre de ressources mises en dominance et la valeur finale de la fonction objectif. En effet, on constate dans beaucoup de cas que moins il y a de ressources mises en dominance, meilleure est la solution rendue. Ceci ne peut vraiment s'expliquer en toute logique et semble être une coincidence puisque l'on serait plutôt amené à penser que le comportement

contraire aurait plus de sens.

Du point de vue du temps de résolution des problèmes, plus on diminue la valeur de *SppModLogItr*, plus les noeuds de branchement prématurément stoppés sont nombreux, moins il y a d'itérations, moins longue est la résolution. On constate que le gain de temps n'est pas négligeable puisqu'on a déjà divisé par trois le temps de résolution optimal qui se trouve dans le tableau (3.1) pour une faible dégradation de la fonction objectif.

Variation de la valeur de *SppModLogCostDecMin* pour une valeur de *SppModLogItr* fixée à 4

Dans le paragraphe précédent, nous avons fait évoluer la valeur du nombre d'itérattions pour le tronçon dans lequel est étudiée la décroissance de la fonction objectif. Il est apparu que la valeur de 4 pour la longueur de ce tronçon parait convenir pour désormais faire varier la valeur de *SppModLogCostDecMin*. Si, pour le noeud de branchement courant, la fonction objectif n'a pas diminué de *SppModLogCostDecMin* au cours des 4 dernières itérations, le modèle cesse et on va donc brancher puisque par défaut seul un modèle unique est défini. Voici les résultats sous forme de tableau des différents jeux de tests qui ont été efféctués :

La légende du tableau (3.4) est comme suit :

Res : Nombre de ressources mises en dominance.

Dom : La dominance sur le noeud puits est-elle effectuée ?

Log : Valeur de *SppModLogItr*.

SppMLCDM : Valeur de *SppModLogCostDecMin*.

Nds : Nombre de noeuds de branchement nécessaires à la résolution.

Itrs : Nombre d'itérations nécessaires à la résolution.

SP : Temps en secondes pour résoudre les sous-problèmes.

PM : Temps en secondes pour résoudre le problème maître.

Tableau 3.4 – Variation de la valeur de *SppModLogCostDecMin* pour une valeur de *SppModLogItr* fixée

Res	Dom	Log	SppMLCDM	Nds	Itrs	SP	PM	SP+PM	FO
1	non	4	50	48	359	620.6	234.2	854.8	43300
1	non	4	70	51	353	593.6	232.5	826.1	43900
1	non	4	90	57	408	686.5	230.8	917.3	44900
1	non	4	100	60	402	665.7	230.1	895.8	45000
1	non	4	150	57	395	693.3	249.8	943.1	46200
1	non	4	200	48	319	545.1	246.3	791.4	43800
1	non	4	500	62	379	619.6	223.3	842.9	46800
2	non	4	50	57	401	754.8	237.3	992.1	44700
2	non	4	70	55	390	720.6	223.5	944.1	43600
2	non	4	90	54	383	719.4	277.9	997.3	43600
2	non	4	100	49	341	642.4	239.2	881.6	43400
2	non	4	150	60	408	743.0	264.3	1007.3	44600
2	non	4	200	57	374	689.8	228.7	918.5	44700
2	non	4	500	47	292	544.3	216.7	761.0	43400
3	non	4	50	53	406	810.1	239.0	1049.1	44000
3	non	4	70	56	398	775.9	225.6	1001.5	44700
3	non	4	90	62	429	819.9	257.0	1076.9	45800
3	non	4	100	59	439	861.3	254.4	1115.7	44900
3	non	4	150	60	403	806.1	250.0	1056.1	45500
3	non	4	200	56	369	715.8	232.7	948.5	46100
3	non	4	450	55	326	639.8	226.5	965.3	43700
4	non	4	50	56	417	1338.7	222.4	1561.1	45500
4	non	4	70	58	400	1259.4	220.5	1479.9	44600
4	non	4	90	56	400	1297.3	240.2	1537.5	44500
4	non	4	100	53	354	1164.0	209.8	1373.8	43400
4	non	4	110	52	349	1194.8	214.9	1409.7	43400

SP + PM : Temps total à la résolution du problème.
FO : Valeur de la solution finale.

Ce qui nous frappe au regard des valeurs du tableau (3.4), c'est l'importance du facteur aléatoire qui intervient aussi bien pour les valeurs finales des fonctions objectif que pour le temps nécessaire de résolution. Il y a bien trop de variance dans les résultats des problèmes. On ne laisse pas les résolutions des noeuds de branchement se compléter et on peut aussi bien se retrouver avec des colonnes intéressantes dans le problème maître que n'avoir que des colonnes pauvres. Une telle approche est difficilement exploitable.

3.2.2 Un critère basé sur le coût réduit des colonnes générées par les sous-problèmes

Dans le prolongement de l'exploration des différents critères dont nous pouvons disposer pour stopper la résolution d'un noeud de l'arbre de branchement ou encore pour changer la valeur d'un paramètre, nous allons nous intéresser aux coûts réduits des colonnes générées par les sous-problèmes.

Calcul du coût réduit moyen des colonnes envoyées dans le problème maître

Lorsque le sous-problème génère des colonnes, chacune d'entre elles a un coût réduit calculé en fonction des colonnes qui sont présentes dans la base du problème maître. Lorsque parmi toutes les colonnes qui sont générées aucune n'a un coût réduit négatif, cela signifie que l'exploration du noeud de branchement courant est terminée puisque la valeur de la fonction objectif ne peut plus décroître. Par contre, si certaines colonnes ont un coût réduit négatif, il est peut-être possible d'améliorer la valeur de cette fonction objectif. Il s'agit donc pour le problème maître d'effectuer

des itérations de simplexe pour savoir quelles colonnes vont entrer en base et quelles colonnes vont en sortir. Pratiquement, le simplexe choisit de faire rentrer d'abord la colonne ayant le coût réduit le plus négatif dans la base car c'est celle qui est le plus susceptible de faire décroître la fonction objectif. La colonne de la base dont le coût réduit est positif en sort et le processus itératif est relancé.

L'idée émise ici est de calculer le coût réduit moyen des colonnes générées par les sous-problèmes, c'est-à-dire le rapport entre la somme de tous les coûts réduits des colonnes générées et le nombre de ces colonnes. En effet, ce coût réduit moyen devrait nous renseigner sur la possibilité de continuer à faire décroître la fonction objectif à un noeud de branchement. Si ce coût réduit moyen est faible (ou grand en valeur absolu car il est négatif), cela signifie que les colonnes générées ont le potentiel de faire changer la valeur de la fonction objectif. Cela revient finalement à dire qu'il existe encore de la richesse à exploiter dans les colonnes non encore générées par les sous-problèmes pour ce noeud de branchement. Si le coût réduit moyen est proche de 0, il est probable que stopper la résolution de ce noeud de branchement n'influencera pas beaucoup la valeur finale de la fonction objectif.

Pour ce faire, dans le module *Spp*, on peut changer le code du programme *lr.c* dans la fonction *fAddCols* pour calculer le coût réduit moyen de toutes les colonnes entrantes dans la problème maître et arrêter la résolution du noeud de branchement si le coût réduit moyen est trop faible en valeur absolue.

Nous allons réutiliser le problème des sections précédentes à quatre ressources mais n'allons effectuer que des tests où seule une ressource est mise en dominance et où aucune dominance n'est effectuée sur le noeud puits. À l'optimalité, la fonction objectif

Tableau 3.5 – Rappel des valeurs rendues pour la résolution à une ressource sans dominance

Nb. ress.	Dom.	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	482	48	898.2	211.2	1109.4	43400

a pour valeur 43300. La dominance sur une ressource est donc plutôt intéressante puisqu'elle ne nous fait perdre seulement qu'un quart de pourcent.

Nous allons faire varier la valeur du coût réduit moyen minimum que les colonnes

entrantes dans le problème maître doivent satisfaire et observer les modifications dans le comportement de la fonction objectif et du temps de résolution du problème. Si ce coût réduit moyen minimum n'est pas satisfait, la résolution du noeud de branchement courant cesse et on explore le noeud de branchement suivant.

Tableau 3.6 – Étude du temps de résolution et de la valeur finale de la fonction objectif en fonction du coût réduit minimum

Nb. ress.	Dom.	crm	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	0	482	48	898.2	211.2	1109.4	43400
1	non	25	615	51	1079.9	215.7	1295.6	43900
1	non	50	344	48	612.6	224.6	837.2	43600
1	non	60	309	50	530.1	213.6	743.7	43600
1	non	75	331	50	566.6	210.7	777.3	43700
1	non	90	384	57	638.9	194.9	833.8	43800
1	non	100	377	56	629.8	207.1	836.9	43400
1	non	125	279	51	470.1	204.2	674.3	43700
1	non	135	340	58	554.8	183.9	738.7	43700
1	non	150	348	60	578.1	191.1	769.2	45200
1	non	160	337	59	546.2	194.2	740.4	45700
1	non	175	272	57	452.8	214.9	667.7	44200
1	non	200	349	61	572.3	192.4	764.7	46500
1	non	250	338	62	540.5	218.8	759.3	46300

La légende de ce tableau vient comme suit :

crm : coût réduit moyen minimum à satisfaire.

Nb Itr : Nombre d'itérations nécessaires pour résoudre le problème.

Noeuds : Noeuds de branchement nécessaires pour résoudre le problème.

Au vu des résultats du tableau (3.6), il apparaît que le critère d'arrêt basé sur le coût réduit moyen est assez efficace et ce pour plusieurs raisons. Tout d'abord, les valeurs finales de la fonction objectif croissent avec le seuil du coût réduit moyen, ce qui peut nous laisser penser que le facteur aléatoire n'est peut-être pas trop influent avec ce critère. Les valeurs de la fonction objectif restent acceptables pour des coûts

réduits moyens inférieurs à 150, l'erreur maximale tourne autour d'un pourcent. Les temps de calculs sont particulièrement bons puisqu'on parvient à les faire chuter d'environ 30 à 40 % par rapport à la dominance sur une ressource.

Changement statique de la valeur du coût réduit moyen minimum

L'idée développée maintenant consiste à changer de façon statique la valeur du seuil du coût réduit moyen pendant la résolution du problème sous GENCOL, un peu comme si deux modèles étaient utilisés. L'idée serait de prendre un seuil plutôt élevé en valeur absolue au début puis plus faible dans un second temps. On délimiterait ainsi les frontières du polygone covexe des solutions réalisables sur lequel nous travaillons assez rapidement. Pour simplifier le changement dans le code de GENCOL, nous allons donner une valeur aux 150 premières itérations et une autre valeur aux itérations suivantes. Les résultats de ces tests apparaissent dans le tableau (3.7). Des résultats obtenus, on peut aisément proposer quelques commentaires. Tout d'abord, il apparait une certaine décroissance de la fonction objectif en fonction des valeurs de plus en plus petites des seuils des coûts réduits moyens pour la seconde partie du problème. Cela est de bonne augure puisque cela montre de façon assez claire que la variance n'a plus autant d'importance dans cette façon de procéder en opposition à la décroissance de la valeur objectif sur un tronçon d'itérations comme critère. Il est donc dorénavant possible d'envisager certaines théories dynamiques pour la résolution des problèmes sous GENCOL.
Par contre, le temps varie quant à lui de facon plutôt aléatoire, on peut avoir de très bons résultats avec des contraintes asses fortes.

Tableau 3.7 – Étude du temps de résolution et de la valeur finale de la fonction objectif en fonction de deux valeurs différentes pour le coût réduit minimum

Nb. ress.	Dom.	crm1	crm2	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	125	25	377	51	650.8	213.1	863.9	43600
1	non	150	25	402	55	752.0	201.9	953.9	43600
1	non	200	25	312	51	538.7	201.3	740.0	43900
1	non	125	50	436	55	730.5	198.5	929.0	43600
1	non	150	50	319	53	528.4	188.5	716.8	43600
1	non	175	50	309	52	503.2	197.3	700.5	43800
1	non	200	50	238	47	394.7	188.6	583.3	43900
1	non	225	50	303	52	505.5	234.8	740.3	44200
1	non	250	50	491	59	854.7	221.2	1074.9	44800
1	non	125	75	255	48	421.2	202.1	623.3	43600
1	non	150	75	348	56	583.2	193.2	776.4	43800
1	non	175	75	381	56	656.9	210.0	866.9	44200
1	non	200	75	331	57	543.6	188.7	732.3	44000
1	non	150	100	327	57	551.4	195.4	746.8	43800
1	non	175	100	319	58	512.9	194.8	707.7	44300
1	non	200	100	303	54	527.7	197.5	724.2	44000

Somme des coûts réduits comme critère

Plutôt que de prendre la moyenne des coûts réduits, nous allons maintenant prendre la somme des coûts réduits des colonnes générées par les sous-problèmes à une itération donnée. Cela nous permet en quelques sortes de prendre en compte le nombre de colonnes générées par le sous-problème et d'en faire une combinaison avec les coûts réduits. Plus le nombre de colonnes est élevé, plus la somme des coûts réduits devrait être élevée (tout ceci en valeur absolue naturellement), plus on prolonge la recherche d'une meilleure solution sur le noeud de branchement. Le tableau (3.8) résume les résultats obtenus.

On remarque que les temps sont de quelque peu supérieurs à ceux obtenus lorsque le critère portait sur la moyenne des coûts réduits. Cependant, on peut constater la large plage de valeurs pour lesquelles les résultats sont assez satisfaisants. Cette technique pourrait être intéressante si l'on n'est pas capable de trouver de façon précise

Tableau 3.8 – Étude du temps de résolution et de la valeur finale de la fonction objectif en fonction de la somme des coûts réduits minimums

Nb. ress.	Dom.	$\sum$ cr	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	150	487	55	878.9	249.1	1128.0	43400
1	non	200	428	51	739.2	213.1	952.3	43700
1	non	250	402	51	690.4	210.4	900.8	43500
1	non	300	468	59	788.0	213.4	1001.4	44600
1	non	350	434	51	766.2	235.9	1002.1	43400
1	non	400	411	54	695.7	224.8	920.5	44600
1	non	450	321	49	542.9	225.5	768.4	43400
1	non	475	372	55	639.9	233.2	873.1	43500
1	non	500	323	47	554.7	226.4	781.1	43400
1	non	525	350	51	611.1	238.2	849.3	43600
1	non	550	313	50	522.8	221.2	744.0	43600
1	non	575	404	57	710.7	238.9	949.6	43500

une valeur seuil intéressante pour stopper la résolution d'un noeud de branchement.

Coût réduit minimum comme critère

Nous allons désormais utiliser comme critère d'arrêt la valeur du plus petit coût réduit (ou plus grand en valeur absolue) sur l'ensemble des colonnes générées par le sous-problème. Nous pouvons estimer que si une colonne a un coût réduit très négatif, il y a toujours de la richesse à puiser dans les colonnes non encore générées par les sous-problèmes et il est dans notre intérêt de pousser plus loin la résolution du noeud de branchement courant. Le tableau (3.9) résume les résultats obtenus.
Les valeurs rendues par les jeux de tests ne laissent pas entrevoir de bonne perspective quant à ce critère d'arrêt. Les temps ne sont pas améliorés de façon conséquente et la valeur finale de la fonction objectif se détériore vite. Un facteur aléatoire assez important entre apparemment en jeu lors de l'utilisation de ce critère.

Tableau 3.9 – Étude du temps de résolution et de la valeur finale de la fonction objectif en fonction de la valeur miniale des coûts réduits des colonnes générées par les sous-problèmes

Nb. ress.	Dom.	crmin	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	50	476	49	861.3	232.5	1096.8	43600
1	non	75	476	53	865.2	254.2	1119.4	43700
1	non	90	382	50	667.1	216.0	883.1	43800
1	non	100	392	52	677.3	227.1	904.4	43400
1	non	125	376	52	675.7	236.7	912.4	44400
1	non	150	372	54	642.5	236.7	879.2	44200

L'écart-type des coûts réduits comme critère

L'écart type pour une série de points x_i avec $i \in [1..n]$ représente la dispersion des points autour de leur moyenne. L'écart type vaut :

$$E = \sqrt{(\sum_{i=1}^{n} x_i - X)^2}$$

avec $X = \frac{1}{n} * \sum_{i=1}^{n} x_i$.

On peut penser que plus l'écart type des valeurs des coûts réduits des colonnes renvoyées par les sous-problèmes au problème maître est élevé, plus il y a encore de la richesse à extraire du problème du fait de l'hétérogénéité des valeurs. On a donc envisagé deux types de tests, dans un premier temps on choisirait comme critère la moyenne des coûts réduits sommée à l'écart type des valeurs. Dans un second temps, prendre l'écart type seulement comme critère d'arrêt.

Pour prendre en compte la dispersion des valeurs des coûts réduits générés par le sous-problème, ainsi que la moyenne de ces coûts réduits, il s'agit donc d'utiliser comme critère d'arrêt une fonction du type $f = \sum cr + k * \sigma$ où σ représente l'écart-type. Une telle fonction nous permet de prendre en compte les deux valeurs avec un facteur multiplicatif. Pour les tests, nous avons fixé la valeur de k à 1. Le tableau (3.10) résume les valeurs obtenues.

Les résultats obtenus sont satifaisants, le temps est réduit de façon significative et

Tableau 3.10 – Étude du temps de résolution et de la valeur finale de la fonction objectif en fonction de l'écart-type et de la somme des coûts réduits

Nb. ress.	Dom.	crmoy+et	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	100	462	55	808.8	228.6	1037.4	43700
1	non	125	343	49	594.1	223.6	817.7	43800
1	non	150	407	54	703.1	215.7	918.8	43300
1	non	175	393	53	685.6	226.8	912.4	43500
1	non	200	343	51	584.7	226.2	810.9	43800
1	non	250	366	58	612.1	220.6	832.7	44600

la fonction objectif croît avec la valeur dela fonction critère. Ce paramètre semble ne pas être trop imprégné d'un facteur aléatoire.

Plus on se rapproche de la fin de la résolution d'un noeud de branchement, plus l'écart type a tendance à baisser, il est donc possible de l'utiliser comme critère d'arrêt. Le tableau (3.11) résume les résultats obtenus.

Une fois encore les résultats sont intéressants et peuvent être exploités pour des po-

Tableau 3.11 – Étude du temps de résolution et de la valeur finale de la fonction objectif en fonction de l'écart-type seulement

Nb. ress.	Dom.	et	Nb Itr	Noeuds	SP	PM	SP+PM	FO
1	non	50	421	54	713.0	224.6	937.6	43500
1	non	75	390	51	691.4	231.9	923.3	43300
1	non	100	405	53	720.4	216.8	937.2	43400
1	non	125	348	54	618.1	231.0	849.1	43400
1	non	150	381	51	688.3	245.6	933.9	43700
1	non	175	349	56	594.5	227.0	821.5	44400

litiques dynamiques d'accélération de résolution. L'expertise va nous révéler quelles sont les valeurs qui peuvent être utilisées comme valeurs seuils pour certaines familles de problèmes.

3.3 Les paramètres de branchement

La résolution d'un problème sous GENCOL débute par le calcul de la solution de la relaxation linéaire du problème. La valeur de la relaxation linéaire va constituer une borne inférieure à la solution du problème puisqu'on ne peut considérer de variables fractionnaires pour la solution recherchée en nombres entiers. En effet, imposer des conditions d'intégrité impliquent des contraintes supplémentaires qui vont faire augmenter la valeur finale de la fonction objectif.

Après le calcul de cette relaxation linéaire, la recherche d'une solution entière commence. À chaque noeud de branchement, il s'agit de prendre un certain nombre de décisions, ce qui constitue une stratégie de branchement. Ces stratégies peuvent être soient optimales, soit heuristiques.

En voyant ce procédé comme une arborescence où le noeud 0 représenterait le calcul de la relaxation linéaire, GENCOL va donc créer deux noeuds fils où il aura imposé une décision pour le premier noeud et pour le second noeud, on prend la décision complémentaire afin d'assurer l'optimalité en traitant tous les cas. La figure (3.3)

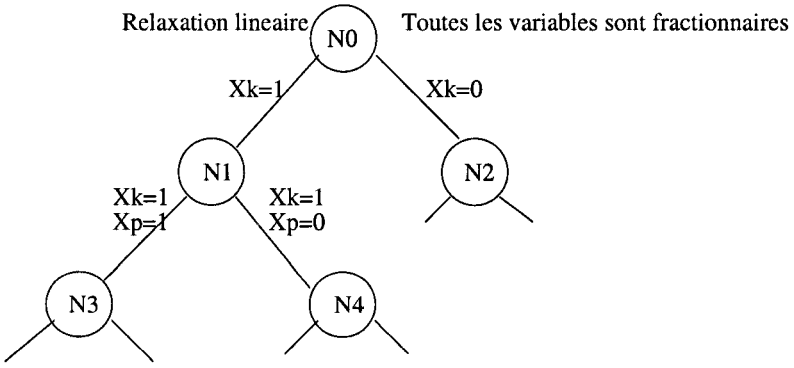

Figure 3.3 – Exemple de branchement sous GENCOL

nous représente cette arboresence où chaque noeud possède deux noeuds fils. On a fixé une variable à 1 pour le premier noeud et à 0 (la valeur contraposée) pour le second noeud. Pour chaque noeud, la valeur de la solution trouvée pour son noeud père constitue une borne inférieure puisqu'on rajoute une contrainte d'integrité.

Dans l'implémentation de GENCOL, il existe trois différents parcours de cet arbre de branchement qui sont :

- Profondeur d'abord (par défaut)

  GENCOL va à chaque branchement choisir d'explorer le meilleur noeud fils, s'enfonçant ainsi dans l'arborescence. Les avantages de cette recherche sont de trouver rapidement une solution réalisable qui servira de borne inférieure pour une nouvelle recherche de l'arbre.
- Meilleur d'abord

  À chaque itération, on explore les fils du noeud qui a jusque là la meilleure solution. Les avantages de cette recherche résident dans le fait qu'on a potentiellement plus de chances de tomber directement sur la (ou une) bonne solution. Un inconvénient de cette recherche est le gros besoin de mémoire puisqu'on explore beaucoup de noeuds avant d'atteindre une solution réalisable.
- Profondeur d'abord puis meilleur d'abord

  Cette recherche se caractérise par une recherche en profondeur d'abord, qui va nous rendre une valeur d'une solution réalisable qui va servir de borne inférieure pour ensuite rechercher en meilleur d'abord afin de limiter le besoin de mémoire. Cette solution peut constituer un bon compromis.

Le choix de la stratégie d'exploration

Comme cela a été expliqué dans le paragraphe précédent, il existe trois manières différentes de parcourir notre arbre de branchement. Par défaut, GENCOL utilisera une recherche en profondeur d'abord jusqu'à trouver une solution réalisable. Cette méthode est intéressante puisqu'elle va nous donner dans des délais assez brefs une borne supérieure qui va nous permettre d'éliminer certains noeuds de branchement. Une recherche en meilleur d'abord devrait rendre un premier résultat réalisable plus intéressant mais dans des délais plus longs et avec une consommation de mémoire plus importante.
Le choix de la méthode de recherche peut dépendre en fait de la taille du problème du point de vue du nombre de tâches. Il devrait être plus intéressant pour un problème ne comportant pas trop de tâches à couvrir d'utiliser une recherche en meilleur

d'abord alors que pour un très gros problème, une recherche en profondeur d'abord s'avèrera probablement être plus intéressante.
La troisième méthode d'exploration de l'arbre de branchement, c'est à dire, profondeur puis meilleur d'abord devrait convenir à des problèmes de taille intermédiaire. On pourrait aussi mettre l'accent sur le choix de la méthode de recherche par rapport au type de problème à résoudre. S'il s'agit d'un problème où il faut trouver une solution coûte que coûte dans les plus brefs délais, qu'elle soit bonne ou mauvaise, il est évident qu'une recherche en profondeur d'abord est l'outil adapté à la situation. Pour finir, il est possible qu'une alternance entre profondeur d'abord et meilleur d'abord puisse rendre des résultats très acceptables.

La méthode de branchement Cfix

Cfix est une méthode de branchement qui consiste à fixer une (ou plusieurs) colonne après la résolution de la relaxation linéaire du noeud courant engendrant ainsi un et un seul noeud fils. Une version modifiée de Cfix permet de créer deux noeuds fils, l'un avec la valeur de la colonne fixée et l'autre avec les chemins devenus interdits du fait de cette colonne fixée. Fixer une colonne signifie affecter un chemin à une commodité. Une fois une colonne fixée, on élimine l'ensemble des tâches couvertes par cette colonne pour la suite du calcul.
Par défaut, Cfix va fixer une et une seule colonne, celle dont le flot aura la plus grande partie décimale. Cfix va donc fixer cette colonne à la valeur du premier entier supérieur à la valeur du flot.
Il existe trois paramètres dans Cfix qui permettent d'influencer certaines décisions de l'arbre de branchement :

- CfixSelectMin décide du nombre minimum de colonnes à fixer d'un noeud de branchement à l'autre.

- CfixSelectMax décide du nombre maximum de colonnes à fixer d'un noeud de

branchement à l'autre.

- CfixSelectThreshold décide le seuil minimum à avoir pour pouvoir fixer la colonne.

Les paramètres CfixselectMin et CfixSelectMax sont prioritaires par rapport au paramètre CfixSelectThreshold dans la mesure où seront fixées d'un noeud à l'autre au moins CfixselectMin colonnes et au plus CfixSelectMax même si le seuil minimal n'est pas respecté.

Modification de CfixSelectMin et CfixSelectMax

Par défaut, la méthode de branchement Cfix ne fixe qu'une colonne au passage d'un noeud à l'autre et CfixSelectMin=CfixSelectMax=1.
En modifiant la valeur de ces paramètres à k par exemple, on va donc fixer k colonnes d'un noeud à l'autre. Ces k colonnes vont recouvrir un certain nombre de tâches que GENCOL va marquer comme couvertes. Cela va aboutir à un nombre de noeud de branchement inférieur à celui qu'on aurait obtenu avec k=1 et probablement une accélération relative de la restitution d'une solution par GENCOL.
Les avantages de cette technique ne résident que dans le gain de temps obtenu car il parait clair que la valeur de la fonction objectif finale va être détériorée. En effet, en fixant une colonne, on attribue finalement un chemin à une commodité qui va couvrir un certain nombre de tâches. Pour le noeud de branchement courant, ces colonnes sont les mieux adaptées pour couvrir ces tâches mais ce n'est pas forcément le cas à l'optimum. Il s'en suit une détérioration de la fonction objectif qui peut-être assez dramatique dans le cas où GENCOL ne parvient pas à trouver une solution parce que l'on fixe trop de colonnes d'un noeud à l'autre.
À titre d'exemple, le tableau (3.12) nous permet de comprendre grâce à un jeu de tests l'évolution du branchement si le nombre de colonnes à fixer d'un noeud à l'autre diffère. Ces jeux de tests ont été effectués pour un problème où seule une ressource

sur quatre a été mise en dominance et où la dominance n'est pas calculée pour le noeud puits. Au vu des résultat, il faut effectivement de moins en moins de noeuds

Tableau 3.12 – Jeu de test avec la modification de CfixSelectMin

CfixSelectMin	Nb de noeuds	nb d'itérations	Tps SP	Tps PM	Temps	Objectif
1	54	595	711.3	157.4	868.7	43300
2	24	465	584.8	126.5	711.3	43800
3	17	335	423.9	115.4	539.3	43400
4	13	427	571.5	115.0	686.5	44500
5	10	361	491.3	116.6	607.9	43300
6	18	435	550.4	102.7	653.1	43300
7	7	275	360.8	97.7	458.5	43700
8	6	273	356.6	99.9	456.5	45000
9	7	389	499.0	96.8	595.8	44900
10	6	348	459.9	101.3	561.2	44500
15	4	315	505.6	113.7	619.3	44200
20	4	336	521.6	97.0	618.6	45400
25	2	143	197.3	84.5	281.8	INF

de branchement pour résoudre les itérations, la fonction objectif est de plus en plus détériorée et GENCOL peut même ne pas trouver de solution réalisable, tout le temps consacré au calcul est ainsi perdu. Il est indéniable que le temps pour rendre une solution réalisable est inférieur à celui du cas par défaut mais il apparait quand même un facteur aléatoire assez important dans cette valeur.

Modification du seuil

Par défaut dans GENCOL, la valeur du CfixSelectThreshold n'est pas utilisée puisqu'on fixe une et une seule colonne, celle qui a le meilleur seuil, c'est à dire celle dont la partie décimale est la plus grande. On peut donc utiliser la valeur de CfixSelectThreshold pour fixer plus d'une colonne au passage d'un noeud de branchement à l'autre. En effet, si l'on fixe CfixSelectMin=1 pour imposer qu'au moins une colonne sera fixée à chaque branchement et CfixSelectThreshold=0.95 alors toutes les colonnes dont le seuil sera supérieur à 0.95 seront fixées. Ces colonnes étant potentiellement intéressantes, cela peut s'avérer avantageux de les fixer. Si aucune colonne

n'a un seuil supérieur à 0.95, GENCOL fixera celle qui a le meilleur seuil même s'il est inférieur à CfixSelectThreshold puisque CfixSelectMin, prioritaire sur CfixSelectThreshold impose qu'au moins une colonne soit fixée.

La principale différence entre les deux méthodes qui permettent de fixer plusieurs colonnes lors d'un branchement réside dans la force de la première méthode (utilisant CfixSelectMin=k) qui va prendre les k premiers seuils et fixer les colonnes correspondantes, même si un des seuils est inférieur à 0.5. Cela revient à chercher un résultat coûte que coûte quelque soit le prix à payer. Cette seconde méthode, qui se base sur le seuil, est peut-être moins contrôlable mais probablement plus intéressante puisque si on ne peut pas décider du nombre de colonnes fixées, on peut contrôler la qualité de ces colonnes.

Le tableau (3.13) résume un jeu de test où la valeur du seuil CfixSelectThreshold est modifiée. On peut y voir entre autres le nombre de noeuds explorés et le nombre moyen de colonnes fixées à chaque noeud.

Les résultats de ces tests sont assez intéressants puisque l'on voit que pour des valeurs encore proches de 0.5, la valeur de la fonction objectif finale n'est pas détériorée, ce qui signifie que la fixation de colonnes de qualité est un bon moyen de ne pas détériorer les résultats tout en accélérant le problème.

Tableau 3.13 – Jeu de test avec la modification de CfixSelectThreshold

Score	Noeuds	Moy. Col. fixees	nb d'itér.	Tps SP	Tps PM	Temps	Objectif
0.9	48	1.02	670	802.1	148.4	950.5	43300
0.8	45	1.11	620	712.5	143.2	855.7	43300
0.7	39	1.36	597	692.6	132.0	824.6	43300
0.6	18	2.89	435	550.4	102.7	653.1	43300
0.55	10	5.3	353	485.9	110.5	596.4	43300
0.5	4	13.0	276	364.7	93.3	468.0	45500

Chapitre 4

Le contrôleur dynamique

Dans ce chapitre, nous allons montrer comment exploiter informatiquement les jeux de tests et les conclusions tirés du chapitre précédent à travers la définition d'un contrôleur dynamique.

4.1 Nécessité d'un contrôleur dynamique

Comme cela a été expliqué dans les chapitres précédents, l'objectif de cette maîtrise recherche consiste à trouver un moyen d'accélérer le temps de résolution du logiciel GENCOL afin de résoudre des problèmes de taille importante en un temps limité. La conséquence de cette accélération se fait ressentir au niveau de la valeur finale de la fonction objectif qui peut se détériorer.

Afin d'obtenir un temps de résolution plus bref pour un problème donné, nous disposons des différents paramètres étudiés dans le chapitre 3. Ces paramètres peuvent être rangés dans trois catégories, les paramètres de sous-problèmes, les paramètres de branchement et les paramètres basés sur des critères d'arrêt. Ces paramètres influencent le temps de résolution d'un problème de façon diverse. L'influence de chacun de ces paramètres sur le déroulement de la résolution d'un problème sous GENCOL étant désormais connue, il devient possible d'adapter à la situation courante le bon paramètre.

Le problème réside dans le fait que GENCOL est un programme statique qui ne peut être interrompu en cours de résolution par un utilisateur afin de modifier

la valeur de certains paramètres. La seule solution dont dispose cet utilisateur pour accélérer le temps de résolution est de se construire différents modèles avec des paramètres basés sur des critères d'arrêt qui permettront d'accélérer la résolution si certaines conditions ne sont pas respectées. La construction de ces modèles ne peut se faire qu'avant le lancement de la résolution et il n'est donc pas possible de connaître les grandes lignes de son déroulement au moment de la confection des différents modèles. Il en résulte qu'une telle accélération est assez hasardeuse car elle ne permet absolument pas de contrôler le temps de résolution et peut fortement détériorer la valeur finale de la fonction objectif.

Une solution consiste à inclure dans le code de GENCOL un contrôleur dynamique capable d'estimer l'avancement de la résolution, de comparer celle-ci à la limite de temps alloué et, le cas échéant, de modifier la valeur de certains paramètres pour se rapprocher de l'objectif. On peut mettre l'accent ici sur l'aspect dynamique du contrôleur. En effet, l'utilisateur n'aura pas à apprécier la situation au cours de la résolution pour effectuer des changements, c'est le rôle du contrôleur d'effectuer cette tâche afin de prendre les décisions adéquates sur la valeur des paramètres. Pour un problème donné, la résolution sur deux machines de puissance différente va donc possiblement produire deux solutions différentes, le contrôleur devant, si nécessaire, prendre des décisions plus drastiques pour la machine la plus lente.

4.2 Fonctionnement du contrôleur dynamique

Nous allons maintenant expliquer l'algorithme sur lequel repose le contrôleur dynamique.

4.2.1 Estimation du temps de résolution

Avant de prendre la moindre décision sur la valeur des paramètres, il est nécessaire de pouvoir connaître, à tout moment, l'état d'avancement de la résolution. État d'avancement signifie ici de l'information sur cette résolution (le numéro de l'itération courante, le temps écoulé jusqu'à présent...) mais surtout le temps estimé pour terminer le problème. L'obtention de cette valeur est assez délicate puisqu'elle est, entre autres, fonction du nombre de noeuds de branchement que l'on ne peut prédire à l'avance. Pour la relaxation linéaire courante, il est possible d'extrapoler la fonction exprimant la valeur de la fonction objectif par rapport au temps par une exponentielle et ainsi estimer le temps de cette relaxation linéaire. Toutefois, il n'est pas possible de prédire le nombre de noeuds de branchement ainsi que l'allure des relaxations linéaires des futurs noeuds de branchement. Cette difficulté nous pousse à trouver un critère qui ne dépende pas du nombre de noeuds de branchement pour effectuer notre extrapolation.

La seule manière qui permettrait vraiment d'extrapoler le temps de résolution d'un problème devrait être basée sur la décroissance du nombre de variables fractionnaires. En effet, dans la plupart des cas, le nombre de variables fractionnaires est maximal une fois la relaxation linéaire du problème trouvée, puis il décroît au fur et à mesure des divers branchements effectués de manière assez continue jusqu'à zéro. Lorsque le nombre de variables fractionnaires atteint la valeur nulle, celà signifie que la solution actuelle est réalisable. Elle peut certes encore être améliorée si elle n'est pas optimale et si le temps alloué à la résolution du problème n'est pas écoulé mais déjà nous avons une première solution qui peut peut-être convenir à l'utilisateur. La figure (4.1) nous permet d'évaluer quelque peu l'allure de la fonction exprimant le nombre de variables fractionnaires par rapport au temps. Sur ce graphique, chaque point représente une itération de GENCOL. À première vue, cette décroissance semble linéaire malgré la branchement qui pourrait plus ou moins compreomettre cette linéarité. Pour d'autres exemples, il s'est avéré que la décroissance prend une forme légèrement incurvée. Nous allons toutefois considérer

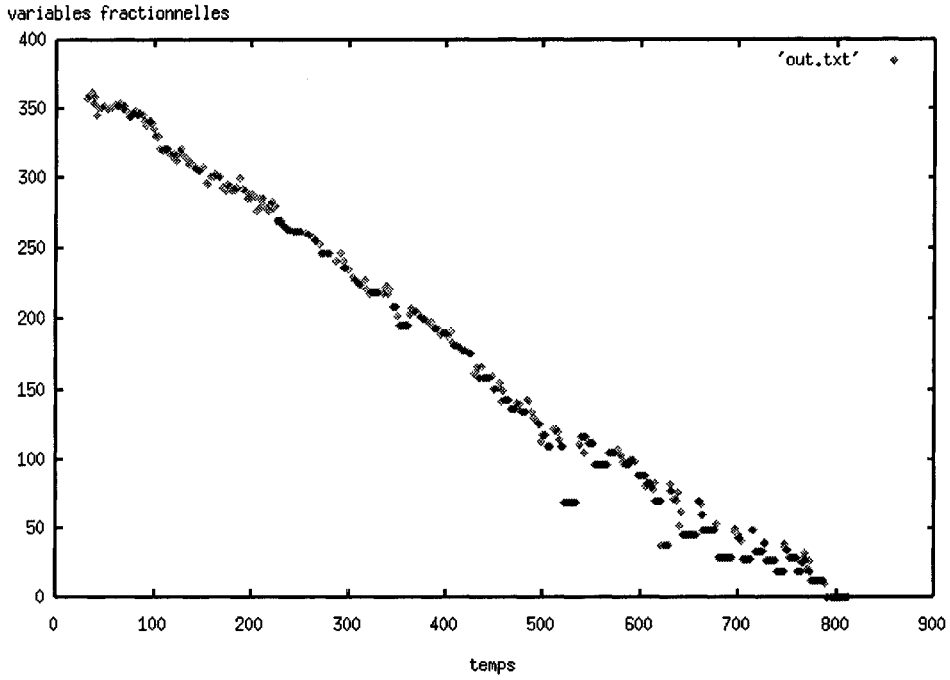

Figure 4.1 – Exemple de décroissance du nombre de variables fractionnaires par rapport au temps.

pour ce mémoire la forme linéaire qui est apparue comme le meilleur compromis pour l'ensemble des problèmes. Une révision de cette décision pour une famille de problèmes donnée peut s'avérer efficace si l'expertise montre que la courbe présente certaines particularités.

Pour extrapoler linéairement un jeu de points composé de N éléments $(x_i, y_i)_{i\in[1..N]}$ par la droite $y = ax + b$, nous utilisons les équations suivantes :

$$a = \frac{N \times \sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} y_i \times \sum_{i=1}^{N} x_i}{N \times \sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2} \tag{4.1}$$

$$b = \frac{\sum_{i=1}^{N} y_i - a \times \sum_{i=1}^{N} x_i}{N} \tag{4.2}$$

Reprenons l'exemple de la figure (4.1) et plaçons nous à l'instant $t = 400$ puis extrapolons. Nous obtenons ainsi une droite et l'intersection de cette droite avec l'axe des abscisses va nous donner le temps estimé pour compléter le problème. La figure (4.2) montre cette démarche et nous permet de constater qu'une extrapolation

linéaire paraît convenir à la situation. La valeur $\frac{-b}{a}$ correspond à l'intersection de la droite extrapolée avec l'axe des abcisses.

Pour choisir les N points qui vont former notre base de calcul pour l'extrapolation,

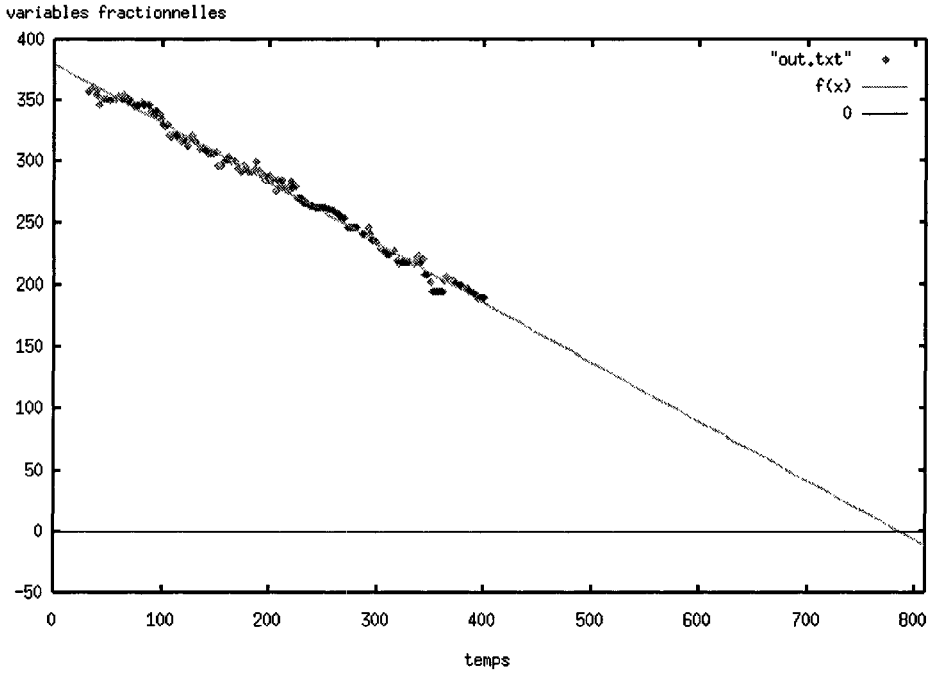

Figure 4.2 – Exemple d'extrapolation du nombre de variables fractionnaires par rapport au temps.

nous aurions pu tout simplement prendre les N premiers points qui correspondent aux N premières itérations de la résolution (ceci en considérant que nous nous trouvons à l'itération N+1), toutefois il faut bien admettre que les points relatifs aux premières itérations n'ont plus beaucoup d'influence sur le temps final de résolution lorsque l'on se trouve proche de la fin. Il y a donc une notion du poids d'un point qui intervient ici et pour résoudre ceci nous avons dicté les règles suivantes :

À l'itération N+1 :

- Si le temps extrapolé à l'itération N est supérieur à deux fois le temps courant, nous considérons les N+1 premiers points pour effectuer notre extrapolation.
- Si le temps t_n extrapolé à l'itération N est inférieur à $2 * t_c$ où t_c représente le temps courant alors ne seront considérés que les points (x_i, y_i) tels que $x_i \geq t_c - (t_n - t_c)$

De cette façon, nous ne considérons que les points qui ont une réelle signification sur

l'extrapolation que nous désirons effectuer.

4.2.2 Appréciation du degré d'avancement

Comme nous l'avons vu dans le chapitre précédent, il existe différents paramètres qui permettent d'accélérer la résolution d'un problème sous GENCOL. Chacun de ces paramètres a un potentiel plus ou moins fort à accélérer cette résolution.
Le problème qui se pose devant nous maintenant consiste à choisir le ou les bons paramètres à modifier pour réagir de façon adéquate à une situation. En effet pour un faible retard sur le temps utilisateur alloué, il ne s'agit pas d'activer, pour un faible lapse de temps, le paramètre ayant le plus gros potentiel à accélérer la résolution. Il faudrait plutôt utiliser un paramètre à potentiel plus doux pendant un temps plus long, garantissant ainsi une perte d'optimalité moins importante. Effectivement, si l'on décide par exemple pour un faible retard d'agir sur le nombre de colonnes fixées par noeud de branchement, il s'agit là d'une forte décision qu'il faudra assumer jusqu'à la fin de la résolution.
On voit donc apparaître ici une notion de degré de retard qu'il faut mesurer pour attribuer une réponse en conséquence. C'est dans cette optique que nous avons mis en place un système de priorités à 5 étages où chaque priorité correspond à la modification de certains paramètres. Ainsi, si dans le courant de la résolution, le temps estimé pour la fin de la résolution est inférieur au temps utilisateur, on se placera en priorité 5 alors que si une retard très important est constaté, on se placera en priorité 1.
Par défaut, la priorité attribuée à la résolution est 5 à chaque itération. Ensuite est effectué le calcul du temps estimé de terminaison du problème puis on vérifie successivement toutes les conditions suivantes pour changer la priorité du problème le cas échéant.
Notons t_e le temps estimé de terminaison, t_c le temps courant et t_u le temps utilisateur.

- Si $t_e > t_u \times 0.95$ alors la priorité donnée est 4.
- Si $t_e > t_u \times 1.15$ alors la priorité donnée est 3.
- Si $t_e > t_u \times 1.35$ alors la priorité donnée est 2.
- Si $t_e > t_u \times 1.55$ ou $t_e < 0$ ou $t_c > t_u \times 0.95$ alors la priorité donnée est 1.

En utilisant ce système de priorités, nous pouvons désormais répondre de façon plus efficace à une situation. La plus forte priorité, i.e la priorité 1, est activée sous trois conditions. La première de ces conditions est relative à un retard excessif qui nécessite une accélération conséquente. La seconde condition est relative à la pente de la droite d'extrapolation du nombre de variables fractionnaires en fonction du temps. En effet, il arrive par moment que le temps extrapolé soit négatif, ce qui signifie que la pente de la droite est positive et que le nombre de variables fractionnaires augmente (ou du moins en moyenne). On s'éloigne donc d'une solution et il faut réagir vigoureusement. Enfin, si le temps utilisateur t_u est à 95 % écoulé, il faut finir rapidement la résolution et on passe donc en priorité 1.

On peut préciser ici le fait que les valeurs seuil proposées pour passer d'une priorité à la suivante peuvent varier d'un type de problème à l'autre, l'expertise permettant de fixer ces valeurs.

Une fois les priorités bien établies, il s'agit de répondre en conséquence avec les paramètres adaptés à la situation. Encore une fois, l'attribution des paramètres à une priorité peut varier d'un type de problème à l'autre si les résultats obtenus sont meilleurs pendant les tests.

La réponse paramétrique aux différentes priorités a été reproduite sur le tableau (5.1) pour un problème à quatre ressources. L'efficacité de cette manière de procéder va être démontrée dans le chapitre suivant où sont effectués les jeux de tests.

4.3 Implantation informatique

Le travail informatique réalisé pour cette maîtrise se répartit dans deux projets différents. Il a fallu tout d'abord implémenter le contrôleur et l'inclure au sein du code

Tableau 4.1 – Valeurs des paramètres en fonction de la priorité

Priorité	5	4	3	2	1
Dominance noeud puits	oui	non	non	non	non
Ressources en dominance	4	3	2	1	1
CoItrLog	0	0	0	5	5
dLogCostDecMin	0	0	0	0	0
CfixSelectMax	1	1	1	100	100
CfixSelectThreshold	0.6	0.6	0.6	0.6	0.6

de GENCOL puis créer une interface graphique pour l'utilisateur afin que celui-ci puisse suivre le déroulement de la résolution.

4.3.1 Le contrôleur

Pour mettre en place le contrôleur, nous avons besoin d'une quantité d'information assez conséquente puisqu'à chaque itération, nous devons connaître le temps écoulé jusqu'à présent et le nombre de variables fractionnaires. Nous devons connaître ces valeurs non seulement pour l'itération courante mais aussi pour les itérations précédentes puisqu'elles nous sont indispensables au calcul du temps extrapolé.
Il faut donc mettre en place une structure informatique capable de garder toute cette information et à laquelle nous pourrions accéder rapidement. Ainsi dans le fichier *spp/struct.h*, nous avons rajouté un type de structure nommé *CtrlItr* qui, comme nous le montre la tableau (4.2), comprend les champs suivants :

- *MpTime* : Ce champ contient un double et représente le temps écoulé pour le problème maître.
- *SppTime* : Ce champ contient un double et représente le temps écoulé pour le sous-problème.
- *FracNb* : Ce champ contient un entier et représente le nombre de variables fractionnaires.

Tableau 4.2 – La structure *CtrlItr*

Type	*Nom du champ*
double	*MpTime*
double	*SppTime*
int	*FracNb*
CtrlItr	*pitrNext*
CtrlItr	*pitrPrev*

- *pitrNext* : Ce champ est un pointeur vers une structure *CtrlItr*.
- *pitrPrev* : Ce champ est un pointeur vers une structure *CtrlItr*.

Les deux pointeurs vers les structures *CtrlItr* nous permettent de former une double liste chaînée. Ainsi pour l'itération *k*, *pitrNext* va pointer vers la structure *CtrlItr* de l'itération *k+1* alors que *pitrPrev* va pointer vers la structure *CtrlItr* de l'itération *k-1*.

Il s'agit maintenant de pouvoir accéder à cette structure pendant le déroulement de GENCOL, pour celà, nous avons lié à la structure déjà existante du sous-problème *SppSPP* une structure de contrôle qui contient les champs du tableau (4.3).

Tableau 4.3 – La structure de contrôle liée à *SppSPP*

Type	*Nom du champ*
double	*Shoot*
int	*Priority*
CO	*coItr*
CtrlItr	*pitrHead*
CtrlItr	*pitrTail*

- *Shoot* : Ce champ contient un double et représente le temps estimé pour achever la résolution du problème.
- *Priority* : Ce champ contient un entier et représente la priorité courante du problème.
- *coItr* : Ce champ contient un entier et représente le nombre de structure *CtrlItr* qui ont été créées.

- *pitrTail* : Ce champ est un pointeur vers le premier élément de la double liste chaînée de structure *CtrlItr*.
- *pitrPrev* : Ce champ est un pointeur vers le dernier élément de la double liste chaînée de structure *CtrlItr*.

Modélisé de cette façon, nous pouvons désormais calculer le temps extrapolé à chaque itération en utilisant un pointeur vers la structure du sous-problème *SppSPP*. De cette structure, nous utilisons un second pointeur pour se rendre sur la première cellule *CtrlItr*. Nous pouvons ainsi accéder aux champs de cette cellule i.e au temps écoulé au moment de l'itération qui lui correspond ainsi qu'au nombre de variables fractionnaires à cet instant. Ensuite en se déplaçant le long de la double liste chaînée, nous pouvons récupérer ces valeurs pour les itérations suivantes et en comptant le nombre de cellules parcourues, il est possible en utilisant les formules énoncées au paragraphe précédent de trouver le nouveau temps extrapolé et éventuellement de modifier la priorité courante du problème.
Lorsque nous ne voulons pas utiliser toutes les cellules pour le calcul de l'extrapolation, nous nous déplaçons le long de la liste chaînée jusqu'à trouver la première cellule qui satisfait nos critères.

4.3.2 L'interface graphique

Toutes les améliorations informatiques ajoutées au sein du logiciel GENCOL pour mettre en place le contrôleur se caractérisent par des petits morceaux de code ajoutés ça et là dans le programme initial. Toutefois, il n'est pas possible pour l'utilisateur de se rendre compte des modifications qu'apporte le contrôleur dans le déroulement de la résolution puisque GENCOL ne produit qu'une sortie en ligne non graphique où sont affichées des informations pratiques sur le déroulement comme le temps écoulé ou le nombre de fractions fractionnaires à l'itération courante.
Pour offrir à l'utilisateur une réelle possibilité de suivre les décisions du contrôleur, nous avons mis en place une interface graphique comme nous le montre le figure (4.3) qui nous permet de suivre en temps réel (i.e pendant l'exécution de GENCOL) le

déroulement de la résolution par l'intermédiaire de la courbe représentant le nombre de variables fractionnaires et la fonction objectif en fonction du temps. De plus, sur notre interface graphique apparaît la droite d'extrapolation ainsi que la valeur estimée de fin de résolution. Parallèlement à la fenêtre représentant les graphiques du déroulement de GENCOL, nous avons ajouté une fenêtre où l'on peut voir la valeur de la priorité courante ainsi que celle des différents paramètres sur lesquels agit le contrôleur.

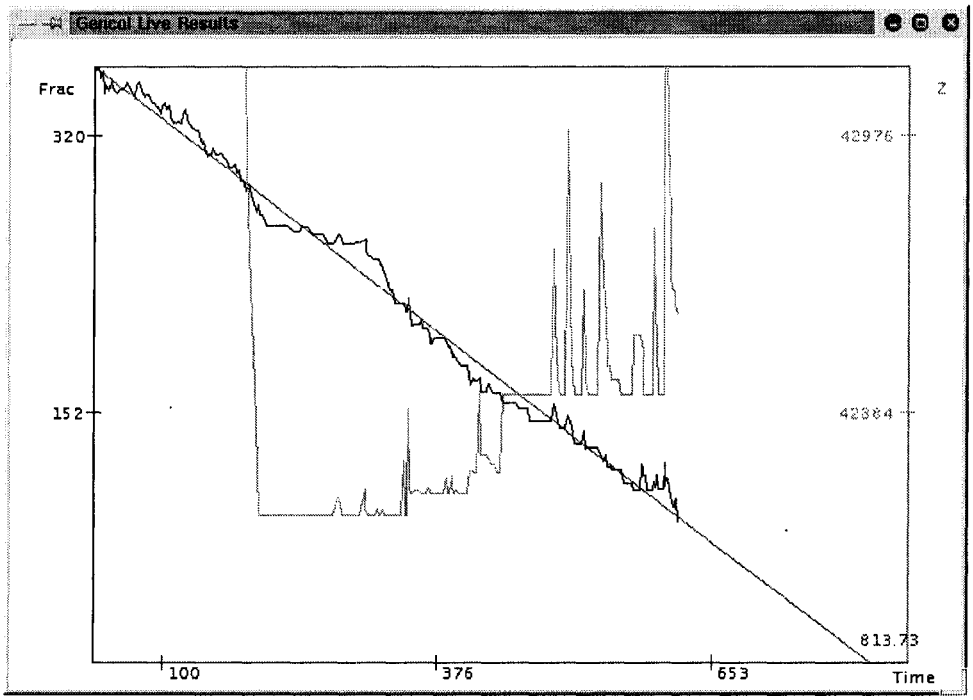

Figure 4.3 – Capture graphique de l'interface utilsateur java.

Pour réaliser cette interface graphique, nous avons utilisé deux langages différents, à savoir Java et PERL. La sortie en ligne de GENCOL est envoyée vers un fichier qui est analysé à chaque fois que sa date de création est modifiée (ce qui signifie qu'il a été complété par de nouvelles itérations). PERL l'analyse et extrait le temps écoulé à chaque itération ainsi que le nombre de variables fractionnaires. Ces informations sont données à Java qui dessine ainsi la courbe en question et l'extrapole linéairement. Parallèlement à ça, GENCOL, lors du déroulement de sa résolution, crée à chaque fois que la priorité est modifiée un fichier qui donne les nouvelles va-

leurs des paramètres qu'il a modifiés. Ce fichier est lu par Java qui crée une fenêtre reproduisant ces informations. Cette fenêtre est représentée à la figure (4.4)

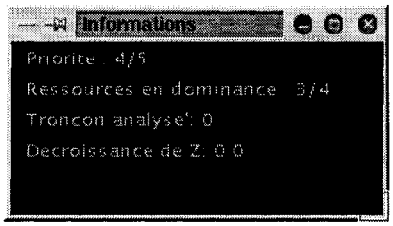

Figure 4.4 – Capture graphique de la fenêtre d'informations de l'interface utilsateur java.

Chapitre 5

Résultats numériques

Nous avons étudié lors des chapitres précédents quelques paramètres qui influent sur le temps de résolution d'un problème sous GENCOL. Nous avons ensuite suggéré la solution d'un contrôleur dynamique pour pouvoir résoudre un problème en un temps prédéfini par l'usager. Des algorithmes ont été proposés pour régir les lois de ce contrôleur et il ne reste maintenant plus qu'à le tester sur des jeux de données.

5.1 Origine des jeux de données

Nous allons tester le contrôleur dynamique grâce au script VCS utilisé au GERAD. Ce script permet de générer des données d'entrée aléatoires au logiciel GENCOL. Il a été conçu pour modéliser un réseau d'autobus qui doit couvrir un certain nombre de tâches. Pour exécuter ce script, il s'agit d'entrer 9 paramètres entiers qui sont les suivants :

vcs a b c d e f g h i

a : Nombre de parcours d'autobus.

b : Nombre de points de relève (1,2,3 ou 4).

c : Constante aléatoire.

d : Coût fixe d'un autobus.

e : Coût fixe d'un chauffeur.

f : Coût relatif d'un autobus.

g : Coût relatif d'un chauffeur.

h : Temps d'attente (ne comprend pas les pauses).

i : Sorties supplémentaires (booléen).

Dans ces paramètres, nous constatons la présence du nombre de points de relève. Celà correspond au nombre d'emplacements sur chaque parcours où les chauffeurs devront être échangés. Le nombre de tâches N à couvrir sur un tel réseau est donné par la formule suivante :

$N = a \times (b + 1)$

En effet, Si un parcours d'autobus comprend b points de relève, il peut être découpé en $b + 1$ tâches à couvrir, chaque tâche représentant le sous-parcours entre deux points de relève.

Ce script construit dynamiquement des réseaux de noeuds et d'arcs qui modélisent le réseau que vont emprunter les chauffeurs. Il s'agit donc de résoudre un problème de plus court chemin avec contraintes de ressources. Ce script traduit ensuite ce graphe généré en fichier d'entrée pour GENCOL. Ce sont ces fichiers qui vont former nos jeux de tests.

Tout au long de nos jeux de tests, nous n'allons faire varier que les deux paramètres a et b qui sont le nombre de parcours d'autobus et le nombre de points de relève. Nous allons faire varier le premier paramètre de 80 à 140 et le second de un à deux. Les problèmes à deux points de relève nécessitant déjà une longue période de résolution, nous n'avons pas cherché à étendre nos jeux de tests à trois points de relève. Les valeurs données aux paramètres non modifiés lors de nos jeux de tests sont les suivantes :

$c = 1$

$d = 50000$

$e = 50000$

$f = 1$

$g = 0$

$h = 0$

$i = 0$

Ces problèmes sont des problèmes à sept ressources qui sont définies comme suit :

DUTY : Durée totale d'une journée de travail pour un chauffeur (incluant briefing, debriefing, déplacement en conduisant un autobus, attente en surveillant un autobus, déplacement sans autobus et pauses).

WORK : Durée totale travaillée dans une journée de travail (ceci inclut tout sauf les déplacements sans autobus et les pauses).
BREAK : Durée maximale d'une pause.
Pour définir les quatre ressources suivantes, il faut comprendre qu'une journée de travail est séparée en pièces de travail. Une pièce correspond au travail (conduite et surveillance) fait sur un même autobus de façon consécutive. Une pause est obligatoire entre deux pièces de travail, ce qui donne :
MINNB et MAXNB : Nombres minimum et maximum de pièces de travail dans une journée de travail
MIN et MAX : Durées minumum et maximum d'une pièce de travail.

5.2 Résultats des jeux de tests

Pour tous les jeux de tests, nous avons utilisé la même présentation.
Chaque ligne représente une résolution où le premier chiffre est l'unique paramètre d'entrée du contrôleur à savoir le temps de résolution requis par l'usager. Les cinq colonnes suivantes numérotées de 1 à 5 représentent le nombre d'itérations effectuées dans les priorités de 1 à 5. Les deux derniers chiffres de la ligne représentent le temps effectif de résolution puis la valeur finale de la fonction objectif.
Pour chaque problème (i.e pour chaque tableau), la première ligne représente la résolution sans l'utilisation du contrôleur pour nous laisser apprécier les différences que peut apporter le contrôleur comparé à la résolution traditionnelle où la méthode de branchement utilisée est Cfix. Nous utilisons l'abréviation OFF comme valeur pour remplir la colonne du temps du contrôleur lorsque celui-ci n'est pas opérationnel. Les paramètres par défaut de GENCOL sont alors utilisés.
La définition des 5 priorités utilisées par le contrôleur est résumée dans le tableau suivant :
À titre de rappel, si la fonction objectif n'a pas décru de dLogCostDecMin sur un

Tableau 5.1 – Valeurs des paramètres en fonction de la priorité

Priorité	5	4	3	2	1
Dominance noeud puits	oui	non	non	non	non
Ressources en dominance	7	4	3	2	1
CoItrLog	0	0	0	5	5
dLogCostDecMin	-	-	-	0	0
CfixSelectMax	1	1	1	100	100
CfixSelectThreshold	-	-	-	0.6	0.6

tronçon de CoItrLog itérations, la résolution du noeud de branchement est stoppée. CfixSelectMax représente le nombre maximal de colonnes que l'on peut fixer au cours d'un branchement et CfixSelectThreshold représente le seuil du score au delà duquel une colonne est fixée.

Voici maintenant les résultats des jeux de tests effectués grâce au script vcs en faisant varier les paramètres a et b du générateur respectivement de 80 à 140 et de 1 à 2.

Tableau 5.2 – Problème à 80 parcours et un point de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	1318	230.5	2904323
200	46	2	12	109	702	171.8	2904323
175	19	1	1	142	556	133.2	2904323
150	20	2	2	171	402	123.3	2904323
125	36	1	0	167	453	107.3	2904323
100	18	0	48	169	365	85.0	2904323
75	19	6	68	125	152	68.0	2954323
50	172	1	15	86	88	54.0	2954327
25	120	16	0	1	67	35.5	2904323

Tableau 5.3 – Problème à 100 parcours et un point de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	2082	1598.6	3705876
1000	30	0	0	196	692	748.5	3655873
750	41	5	44	223	725	682.8	3655926
500	31	1	0	123	171	303.3	3655873
300	33	0	1	212	382	224.5	3655873
250	117	2	2	262	431	241.6	3656004
200	34	41	170	197	167	181.5	3655886
150	59	23	106	98	140	140.90	3655928
100	72	14	52	41	80	99.6	3655891
50	262	0	0	0	57	73.0	3655945

Tableau 5.4 – Problème à 120 parcours et un point de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	3394	4805.6	4156274
2500	27	0	1	345	1072	1979.9	4106227
2250	27	1	0	508	1046	1983.3	4106302
2000	39	0	0	333	669	1584.1	4206135
1750	30	4	54	615	560	1636.2	4106241
1500	44	19	32	279	469	1277.6	4106328
1250	32	19	69	270	475	1127.4	4106336
1000	43	18	36	299	327	924.1	4106289
750	71	2	12	231	174	735.7	4106331
500	194	3	61	103	140	559.7	4106358
400	176	29	115	204	216	400.0	4106187
300	226	6	76	121	168	326.3	4106238
250	111	10	88	97	132	245.3	4106373
100	348	0	0	0	132	214.9	4106334

Tableau 5.5 – Problème à 140 parcours et un point de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	2899	5138.5	4707040
2500	41	0	1	278	784	1897.5	4707040
2250	41	0	1	212	967	1829.9	4707040
2000	41	1	0	202	818	1566.8	4707040
1750	42	0	1	394	523	1526.3	4707040
1500	78	1	148	453	271	1434.6	4807040
1250	45	2	3	394	280	1149.9	4707046
1000	66	7	117	154	89	956.3	4857040
750	173	19	92	94	32	746.6	4757040
500	103	35	26	60	27	479.8	4757040
400	86	13	173	147	54	384.6	4757071
300	64	29	117	59	14	289.0	4707040
100	238	0	0	0	14	218.4	4707040

Tableau 5.6 – Problème à 80 parcours et deux points de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	2347	3249.6	2904323
2500	28	0	0	120	1327	1991.0	2904323
2250	28	0	0	135	1294	1818.9	2904323
2000	28	0	41	371	577	1651.4	2904323
1750	28	0	0	110	1019	1373.5	2904323
1500	53	7	24	293	694	1380.5	2904323
1250	28	1	1	220	470	1125.0	2904323
1000	99	20	126	192	178	981.1	2904323
750	125	1	33	170	169	768.8	2954416
500	51	37	42	114	73	476.5	2904323
400	381	21	54	179	181	435.2	2954323
100	417	0	0	0	120	272.80	2904339

Tableau 5.7 – Problème à 100 parcours et deux points de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	3344	7836.4	3655860
4000	47	0	0	68	1784	2528.1	3705860
3000	54	0	0	244	1375	2258.8	3655860
2500	69	1	5	400	454	2164.0	3655860
2250	138	1	56	478	236	2178.4	3655862
2000	71	33	89	182	204	1843.5	3655860
1750	129	2	154	139	235	1708.9	3655898
1500	173	68	120	46	219	1474.0	3755874
1250	281	93	0	0	178	1413.1	3705871
1000	342	0	0	0	178	1115.4	3705906
750	451	0	0	1	182	1406.6	3706045
500	406	0	0	1	183	1340.9	3705898

Tableau 5.8 – Problème à 120 parcours et deux points de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	¿4000	¿25000	?
10000	50	1	2	368	2183	8577.4	4056170
8000	77	9	42	1164	1503	7250.7	4106195
7000	225	1	133	855	392	6791.4	4156269
6000	86	10	331	213	335	5773.4	4056183
5000	133	94	187	257	262	4919.8	4056342
4000	240	93	87	196	187	4231.3	4056229
3000	337	66	153	32	158	3701.4	4106248
2500	448	18	260	225	235	2669.3	4056320
1000	1013	75	12	0	134	2739.8	4056224

Tableau 5.9 – Problème à 140 parcours et deux points de relève

Tps contrôleur	1	2	3	4	5	Tps final	FO
OFF	0	0	0	0	5199	31367.0	4807055
10000	0	1	0	286	2379	7942.9	4707040
8000	0	1	1	286	1855	6804.1	4707040
7000	4	3	5	533	604	6411.6	4757040
6000	17	3	112	607	185	5714.8	4707040
5000	5	35	98	147	424	4536.3	4707040
4000	50	101	101	129	90	3851.3	4757040
3000	195	62	0	0	130	3462.3	4707042
2500	65	142	80	140	125	2410.2	4707040
2000	369	142	29	2	37	2158.6	4707040
1000	540	0	0	0	0	1957.7	4707040

5.3 Généralités et commentaires sur les résultats

Nous allons maintenant analyser les résultats obtenus sous différents angles à savoir le temps, la valeur finale de la fonction objectif et le nombre d'itérations effectuées dans chaque priorité.

5.3.1 Analyse du temps final de résolution des problèmes

Il y a une relation certaine entre le temps final de la fonction objectif et le temps final exigé par l'usager mais il y a tout de même un certain écart qui peut atteindre parfois jusqu'à 30%.

Il est à noter que le contrôleur n'arrive pas à atteindre des temps trop petits pour la simple raison qu'il existe une borne inférieure au delà de laquelle il n'arrivera plus à descendre. En effet si toutes les itérations sont efectuées avec une priorité 1, le temps necessaire à la résolution du problème devrait constituer une borne inférieure au contrôleur. Si l'usager fixe comme limite un temps inférieur à la borne inférieure, le contrôleur devrait s'exécuter en un temps proche de celui de sa borne inférieure. Ceci peut être observé à chacune des dernières lignes des tableaux de résultats où le

temps usagé est inférieur à la borne inférieure.

Ce problème réapparait également dans les premières lignes des tableaux de résultats où il est à noter que le contrôleur ne parvient pas à atteindre des temps trop grands, proches des temps finaux de résolution en l'abscence de contrôleur. Ceci peut s'expliquer par le fait que pendant le déroulement de la résolution, il est fort probable que l'on passe par une phase où le temps final extrapolé est supérieur au temps final exigé par l'usager. Le contrôleur baisse alors sa priorité pour un certain nombre d'itérations influant directement sur le temps final de résolution. Lorsque le contrôleur repasse en priorité 5, il finira le problème en un temps relativement inférieur au temps de résolution sans contrôleur. D'une façon générale, ces itérations pendant lesquelles le contrôleur baisse sa priorité sont les premières itérations de la relaxation linéaire où le nombre de variables fractionnaires diminue très faiblement voire augmente parfois. Il existe donc également une borne supérieure au delà de laquelle le contrôleur ne pourra plus aller. Cette théorie se confirme en remarquant qu'aucune résolution où le temps usager est grand ne contient toutes ses itérations en priorité 5.

S'il existe un décalage entre le temps usager et le temps final de résolution, il est toutefois à noter qu'il y a une continuité dans les résultats à savoir que les temps finaux de résolution grandissent avec le temps requis par l'usager, ce qui démontre d'une certaine fiabilité du contrôleur et peut-être une révision de certains paramètres dans la définition des priorités précédemment enoncée.

En s'intéressant plus en détails aux résultats, on constate pour la plupart des problèmes résolus qu'il existe une tranche de temps entrées par l'usager pour laquelle les valeurs finales pour le temps de résolution sont particulièrement bonnes.

Prenons le dernier problème résolu (cf. tableau 5.9) et calculons pour chaque ligne du tableau, l'écart entre le temps usager et le temps obtenu et reportons les résultats dans un graphique.

Ce graphique corrobore les hypothèses énoncées précédemment à savoir que le contrôleur éprouve de la difficulté pour des temps relativement petits et des temps relativements grands. Pour des valeurs intermédiaires situées sur le graphique 5.3.1 entre 2000 et 7000 secondes, l'écart entre le temps final de résolution et le temps d'entrée du contrôleur se situe (à une exception près) en dessous de 10 %. Un tel

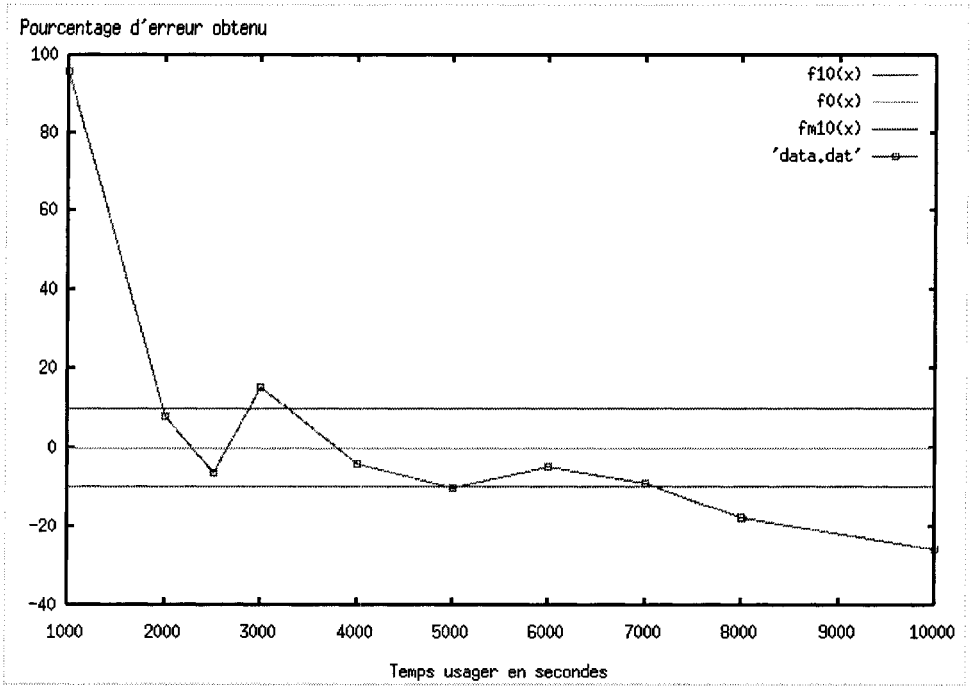

Figure 5.1 – Pourcentage d'erreur entre le temps usagé et le temps final obtenu après résolution

comportement est récurrent dans l'ensemble des jeux de tests que nous avons traités et confirme le bon comportement du contrôleur pour des valeurs intermédiaires de temps d'entrée.

En tentant de généraliser le résultat, nous avons regardé pour chaque jeu de test les première et dernière valeurs du temps pour lesquelles l'écart entre le temps usager et le temps contrôleur est inférieur à 10%. Prenons pour exemple le dernier jeu de test (tableau 5.9), la première valeur pour laquelle l'écart est en dessous de 10% est 2000 secondes et la dernière est à 7000 secondes. Le facteur de réduction du temps de résolution entre le début et la fin de la zone de contrôle avec une erreur inférieure à 10 % est donc 3,5. En faisant la moyenne de tous les problèmes, nous trouvons que le ratio moyen est de 4,6 avec un minimum à 1,5 et un maximum à 10,0. On peut donc déduire de cette analyse que l'on peut en moyenne diviser le temps par un facteur de 4,6 tout en restant proche des temps limites désirés par l'usager. De plus, lorsque l'on prend le temps usager du premier problème qui entre dans la zone des 10% et du dernier problème qui en sort, nous avons trouvé que sur l'ensemble des jeux de tests, 85% des points situés entre ces deux extrémités restaient dans la

bande des 10%.

5.3.2 Valeur finale de la fonction objectif

Les valeurs finales des fonctions objectif sont sujettes à beaucoup de variance. Théoriquement, plus le temps de résolution exigé par l'usager est petit, plus les paramètres utilisés par le contrôleur devront être drastiques, plus la fonction objectif finale devrait être détériorée. Ceci peut s'observer dans les tableaux 5.2, 5.5 et 5.7 où la fonction objectif se dégrade quelque peu avec le temps usager qui diminue. Toutefois, dans la plupart des cas, les valeurs finales de fonction objectif sont très proches de la valeur finale lorsque le contrôleur est désactivé et même quelque fois meilleures (cf. 5.9). Il existe beaucoup de variance dans ce domaine selon les décisions prises par le contrôleur (queues de résolution coupées, nombre de colonnes fixées au branchement) et il est délicat de chercher à définir les lois qui régissent la valeur finale de la fonction objectif.
Cfix n'étant pas une méthode de branchement optimale, on ne peut être sûr de la valeur de la solution optimale de chaque problème mais l'écart entre la meilleure valeur de fonction objectif obtenue et la moins bonne se situe en dessous de 2 à 3 %.

5.3.3 Nombre d'itérations effectuées dans chaque priorité

Il semble clair que plus le temps de résolution exigé par l'usager est grand, plus il y aura d'itérations en priorité 5 et inversement plus le temps exigé par l'usager est petit, plus il y aura d'itérations en priorité 1.
Ceci s'observe aisément dans tous les tableaux où le nombre d'itérations effectuées en priorité 5 diminue de ligne en ligne alors que celui des itérations effectuées en priorité 1 augmente de ligne en ligne. Il y a donc une translation des itérations effectués en priorité 5 vers les priorités inférieures en fonction que le temps usager diminue.

Pour les priorités 2, 3 et 4, le nombre d'itérations effectuées varie de façon parabolique puisque pour des temps elevés, la quasi-totalité des itérations s'effectuent en priorité 5 donc ces priorités ont très peu d'itérations. Plus le temps diminue, plus elles accueillent des itérations qui se déplacent de la priorité 5 vers la priorité 1 et lorsque le temps est très petit, la grande majorité des itérations ont été effectué en priorité 1.

5.3.4 Conclusion sur les jeux de tests

Nous pouvons conclure de ces jeux de tests que le contrôleur permet de fortement diminuer le temps de résolution d'un problème. Cette diminution peut atteindre un facteur 20 (E.g. tableau 5.3) et de façon générale, la dégradation de la fonction objectif est peu significative, de l'ordre de 2 à 3 %. Ces résultats sont donc très encourageants et nous démontrent qu'avec seulement quelques paramètres activés et opérationnels dans le contrôleur, il est possible d'obtenir des résultats très satisfaisants.

Si ces résultats ont été obtenus avec des problèmes d'affectation de chauffeurs à des parcours d'autobus, il n'en demeure pas moins que le contrôleur devrait se comporter pareillement avec des problèmes d'horaires d'équipage puisqu'ils font partie de la même famille de problèmes.

Chapitre 6

Conclusion

Le travail de ce mémoire a consisté à ajouter un contrôleur dynamique au logiciel d'optimisation GENCOL développé au GERAD.

En effet, les compagnies aériennes se doivent de remettre à jour leurs plans d'opération lorsque ceux-ci subissent des perturbations, et ce dans des délais très brefs. La plupart des logiciels d'optimisation qui existent sur le marché permettent à leurs utilisateurs de faire varier le temps de résolution grâce à la modification de certains paramètres d'optimisation internes aux logiciels. Ces paramètres sont souvent fixés avant le début de la résolution et ne peuvent plus être modifiés en cours de résolution. Les utilisateurs ont donc un outil pour faire varier le temps de résolution d'un problème mais ils ne peuvent en aucun cas savoir au préalable le temps que prendra une résolution. Dans ce mémoire, nous avons développé l'approche inverse en faisant varier les paramètres internes de GENCOL à partir d'une limite finale sur le temps de résolution connue.

Pour procéder avec une telle approche, il a fallu dans un premier temps identifier les paramètres internes du logiciel qui pouvaient être modifiés pendant le déroulement de la résolution de GENCOL. On a pu classer ces paramètres en trois catégories à savoir les paramètres du problème maître, du sous-problème et enfin les paramètres de branchement.

Une fois ces paramètres identifiés, des jeux de tests ont été nécessaires pour mettre à jour le comportement de la fonction objectif et du temps de résolution par rapport à une modification de ces paramètres en cours de résolution. Il a ainsi été possible de sélectionner certains paramètres et d'en rejeter d'autres mais surtout de classer les paramètres par le potentiel qu'ils offrent à écourter la résolution d'un problème. Tous ces tests ont été effectués dans un souci de conserver une valeur finale pour la

fonction objectif proche de l'optimum.

Des stratégies ont ensuite été adoptées pour la mise en place du contrôleur dans le code de GENCOL. Il s'agit d'extrapoler le temps de résolution d'un problème en se basant sur la décroissance du nombre de variables fractionnaires. Partant de l'hypothèse réaliste que cette décroissance était linéaire, il a été possible d'extrapoler le temps de résolution en utilisant les algorithmes de Levenberg-Marquardt pendant son déroulement. Une structure en double liste-chaînée a ainsi été codée dans GENCOL pour accueillir à chaque itération les données nécessaires au contrôleur.

En adoptant un système de priorités de 1 à 5 pour le contrôleur selon que le temps extrapolé est supérieur ou inférieur à la limite entrée par l'usager, nous avons fait varier certains des paramètres identifiés précédemment en fonction de ces priorités dans une série de jeux de tests. Ces jeux de tests proviennent d'un petit script générant des problèmes d'horaires de chauffeurs d'autobus qui ont la particularité d'avoir la même structure que les problèmes de mise à jour des horaires d'équipage.

Nous avons pu montrer grâce à ces jeux de tests un bon comportement du contrôleur avec une fonction objectif à peine détériorée et un temps de résolution qui peut être divisé par un facteur 20 dans certains cas. Nous avons montré qu'en utilisant seulement quelques paramètres étudiés, le facteur de réduction du temps de résolution entre le début et la fin de la zone de contrôle avec une erreur inférieure à 10 % est donc 4,6. La dégradation de la fonction objectif est tout à fait acceptable puisqu'elle se situe aux alentours de 2 à 3 %. A également été mise en évidence l'existence de deux bornes inférieure et supérieure, propres à chaque problème qui caractrisent les temps minimal et maximal au delà duquel le contrôleur ne peut pas aller, la zone de contrôlabilité occupant la majeure partie entre ces bornes.

Ce qu'il est important de constater sur la faqn dont a été implémenté le contrôleur, c'est le faible effort requis par l'usager pour obtenir un fonctionnement efficace. En effet, le rôle de l'usager est va se borner à classer les paramètres dans l'ordre de leur influence sur le déroulement de la résolution. Le contrôleur va extrapoler ce temps de résolution et augmenter ou diminuer la priorité suivant qu'on est en retard ou en avance. La force de cette méthode réside dans le fait que le contrôleur va corriger à chaque itération de GENCOL le temps extrapolé pour être de plus en plus précis

puisque le nombre depoints extrapolés va en grandissant. La seule estimation faite dans toute l'approche qui a été utilkisé pour implémenter le contrôleur réside dans cette extrapolation. Sachant que plus le déroulement de la résolution est avancé, plus on est précis sur cette extrapolation, les résultats obtenus par le contrôleur au niveau du temps de calcul ne peuvent être que bons si les paramètres ont été correctement ordonnés. Il existe donc ici une notion d'auto-adaptation du contrôleur qui rend cette méthode robuste quel que soit le jeu de données fourni en entrée à GENCOL.

Les perspectives futures d'un tel contrôleur sont donc bonnes et il est déjà possible de penser à certaines améliorations comme l'ajout de paramètres supplémentaires dans la liste des paramètres modifiées par le contrôleur. Il est également possible d'augmenter ou diminuer le nombre de priorités en fonction de l'expertise sur certaines familles de problèmes. La résolution d'un problème pourrait également commencer dans une priorité intermédiaire plutôt qu'en priorité basse pour minimiser les ajustements à effectuer en début de résolution.

Bibliographie

[1] DANTZIG, G.B. et WOLFE, P. (1960). Decomposition Principle for Linear Progams. Operations Research, 8, 101-111.

[2] DESROCHERS, M. et F. SOUMIS (1988). A Generalized Permanent Labeling Algorithm for the Shortest Path Problem with Time Windows, INFOR 26, 191-212.

[3] GOFFIN, J.L. et VIAL, J.P. (1998). Multiple Cuts in the Analytic Center Cutting Plane Method. Les cahiers du Gerad, G-98-26.

[4] LAVIGNE, J., DESAULNIERS, G., DESROSIERS, J. et SOUMIS, F. (1997). An Introduction to GENCOL. Groupe d'Études et de Recherche en Analyse des Décisions (GERAD).

[5] DESAULNIERS, G., DESROSIERS, J., GAMACHE, M. et SOUMIS, F. (1997). Crew Scheduling in Air Transportation. Les cahiers du Gerad, G-97-26.

[6] PRESS, W.H., FLANNERY, B.P., TEUKOLSKY S.A. et VETTERLING W.T. (1986). Numerical Recipes, The Art of Scientific Computing, Cambridge University Press.

[7] MARQUARDT, D.W. (1963). J. Soc. Ind. Appl. Math., vol. 11, pp. 431-441

[8] DESROSIERS, J., DUMAS, Y., SOUMIS, F., et SOLOMON M.M.(1995). Time Constrained Routing and Scheduling. In M.O. Ball et al., editor, Network routing, Hanbooks in OR, pages 35-139. Elsevier Science, Amsterdam.

[9] STOJKOVIĆ, M., SOUMIS, F. et DESROSIERS, J. (1997). The Operational Airline Crew Scheduling Problem. Les cahiers du Gerad, G-97-12